НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмне забезпечення для захисту авторського права на основі технології Blockchain»

Виконав:
студент IV курсу, групи КП-52
Саприкін Артем Олексійович _____

Керівник:
Старший викладач
Бухтіяров Ю.В. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ПЗКС, к.т.н.,
Вунтесмері Ю.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Саприкін Артем Олексійович
(прізвище, ім'я, по батькові студента)

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКОГО ПРАВА НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

АНОТАЦІЯ

Даний дипломний проект присвячений розробленню web-сервісу для передачі прав контент за допомогою технології блокчейн.

У роботі виконано порівняльний аналіз існуючих рішень для передачі прав контент, проаналізовано методи для передачі прав контент, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даного web-сервісу. Використання блокчейну у данному дипломному проекті забезпечує неможливість змінити попередні записи. Таким чином, система дозволяє користувачеві, який почав розроблювати продукт, а це може бути програмне забезпечення, написання книги тощо, може записувати проміжні єтапи до блокчейну у зашифрованому вигляді. При виникненні плагіату або спірних питань щодо прав власності система їх врегулює, виходячи із дати записаній на блокчейні.

У даному дипломному проекті розроблено: архітектуру серверної та клієнтської частини web-сервісу. Смарт контракт надає можливість відстежувати усі ставки на аукціоні, відслідковувати історію усіх продавців і покупців, а також визначати переможця у децентралізований спосіб. Таким чином, забезпечується можливість дізнатись усю їхню статистику і впевнитись в реальності цих осіб.

Saprykin Artem
(student first name, last name)

SOFTWARE SYSTEM FOR COPYRIGHT PROTECTION BASED ON BLOCKCHAIN TECHNOLOGY

ABSTRACT

This work is devoted to the development of a web-service to transfer content rights via the blockchain.

In this work a comparative analysis of existing solutions for the transfer of rights of content is made, methods for the transfer of content rights are analyzed, the choice of technologies and auxiliary libraries of the server and client parts for the implementation of this web-service are grounded. The use of blockchain in this graduation project makes it impossible to change previous records. Thus, the system allows a user who has started to develop a product, and this may be software, writing a book, etc., can write intermediate stages to the blockade in an encrypted form. In case of plagiarism or controversial issues regarding property rights, the system shall be regulated on the basis of the date recorded on the blockchain.

This graduation project has developed: the architecture of the server and client part of the web-service. Smart contract provides an opportunity to track all bets in the auction, track the history of all sellers and buyers, and determine the winner in a decentralized way. Thus, it is possible to find out all their statistics and to verify the reality of these individuals.

Студент _____ _____
(*підпис*) (*прізвище, ім'я, по батькові*)

Науковий керівник _____ _____
(*підпис*) (*посада, вчене звання, науковий ступінь, прізвище, ініціали*)

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія» (Програмне забезпечення комп'ютерних та інформаційно-пошукових систем)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____ 2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Саприкіну Артему Олексійовичу

1. Тема проекту «Програмне забезпечення для захисту авторського права на основі технології блокчейн», керівник проекту Бухтіяров Юрій Вікторович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз мов програмування та технологій розроблення web-сайтів;
   - розроблення web-ресурсу для захисту авторського права;
   - опис розроблених алгоритмів та підпрограм;
   - аналіз розробленого web-ресурсу.

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних і смарт-контрактів (креслення);
   - діаграма прецедентів (креслення);
   - алгоритм запису, читання і передачі контенту (плакат);
   - структурна схема системи (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури web-ресурсу	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Програмна реалізація web-ресурсу	10.03.2019	
8.	Тестування web-ресурсу	17.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	21.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент А.О. Саприкін

Керівник проекту Ю.В. Бухтіяров

ДП.045440-01-90 Програмне забезпечення для захисту авторського права на основі технології Blockchain. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмне забезпечення	5	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Технічне		
	завдання		
ДП.045440-03-81	Програмне забезпечення	54	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Пояснювальна		
	записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Програма та		
	Матодика тестування		
ДП.045440-05-34	Програмне забезпечення	10	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмне забезпечення	1	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Архітектура		
	системи. UML діаграма		
ДП.045440-07-99	Програмне забезпечення	1	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Алгоритм		
	покупки. UML діаграма		
ДП.045440-08-98	Програмне забезпечення	1	
	для захисту		
	авторського права на		
	основі технології		
	блокчейн. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКОГО ПРАВА НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.О. Саприкін

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для захисту авторського права на основі технології Blockchain.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Сікорського».

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості збереження, продажу та захисту авторського права.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Система повинна забезпечувати такі основні функції:

1) можливість запостити контент у зашифрованому вигляді до блокчейну Ethereum.автоматична класифікація документів за допомогою нейронних мереж із участю користувача;
2) можливість передати права на контент через транзакцію на блокчейні;
3) можливість переглянути історію продажів та покупок певної людини(адреси на блокчейні);
4) Можливість виставити свій контент/профайл для продажу або аукціону;

Розробку виконати на платформі Django з використанням технології блокчейн.

Додаткові вимоги:

1) використання декількох блокчейнів для зберігання контенту;
2) Реалізація програмного API для п.1-4;

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Можливості програмних засобів. Діаграма прецедентів»
   - «Структура бази даних. ERD діаграма»

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту 15.10.2018
Розроблення та узгодження технічного завдання..... 15.11.2018
Підготовка матеріалів першого розділу дипломного проекту 30.11.2018
Розроблення структури web-ресурсу 20.12.2018
Програмна реалізація web-ресурсу..... 08.02.2019
Тестування web-ресурсу 25.03.2019
Підготовка третього розділу дипломного проекту 09.04.2019
Підготовка матеріалів графічної частини проекту 07.05.2019
Оформлення програмної документації 12.06.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКОГО ПРАВА НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.О. Саприкін

2019

ЗМІСТ

ВСТУП

На сьогоднішній день для того, щоб підтвердити права власності на продукт потрібно оформити патент, що буває не завжди зручно через певну низку бюрократичних питань. Також можна використовувати сертифіковані сервіси, які беруть на себе основну частину юридичних питань, але їх недолік полягає у тому, що вони є платними і немає гарантії, що вони будуть продовжувати свою роботу у майбутньому.

При проведенні аукціонів на Ebay або на інших платформах не вистачає прозорості, що ставка була зроблена під час аукціону, а не після. Також неможливо впевнитись, що ставки робить реальна людина, а не робот для накрутки ціни.

Використання блокчейну для вирішення цих проблем є дуже доречним, оскільки неможливо змінити попередні записи. Таким чином, людина яка почала розроблювати продукт, а це може бути програмне забезпечення, написання книги тощо, може записувати проміжні єтапи до блокчейну у зашифрованому вигляді. При виникненні плагіату або спірних питань щодо прав власності завжди можна подивитись на дату постінгу цього контенту в блокчейн.

Для вирішення другої проблеми розподілена база даних допоможе відстежувати усі ставки на аукціоні, відслідковувати історію усіх продавців і покупців, а також визначати переможця у децентралізований спосіб. Таким чином, можна буде дізнатись усю їхню статистику і впевнитись в реальності цих осіб.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Blockchain – це зростаючий список записів, які називаються блоками, які пов'язані за допомогою криптографії. Кожен блок містить криптографічний хеш попереднього блоку тимчасової мітки і дані транзакції (як правило, представлені як дерево Merkle). За конструкцією блокчейн стійкий до модифікації даних. Це відкрита, розподілена книга, що дозволяє ефективно та постійно реєструвати операції між двома сторонами.

Смарт-контракт – це комп'ютерний протокол, призначений для цифрового полегшення, перевірки або забезпечення виконання переговорів або виконання контракту. Інтелектуальні контракти дозволяють здійснювати достовірні угоди без третіх сторін.

ERC-20 – є технічним стандартом, який використовується для смарт-контрактів на блокчейні Ethereum для реалізації токенів, що випускаються на блокчейні Ethereum.

Транзакція – це передача криптовалюти, яка транслюється в мережу і збирається в блоки. Транзакція, як правило, посилається на попередні виходи транзакцій як нові входи транзакцій і перетворює всі вхідні значення на нові виходи. Транзакції не шифруються, тому можна переглядати кожну зібрану транзакцію в блок. Як тільки транзакції наберуть достатню кількість підтверджень, їх можна вважати незворотними.

Адреса криптовалюти – використовуються для надсилання або отримання транзакцій в мережі. Адреса зазвичай представляється як рядок буквено-цифрових символів.

Децентралізована програма (Dapp) – це програма з відкритим вихідним кодом, працює автономно, має свої дані, що зберігаються на блокчейні.

Публічний ключ – загальнодоступною адресою є криптографічний хеш відкритого ключа. Вони діють як електронні адреси, які можуть бути опубліковані в будь-якому місці, на відміну від приватних ключів.

Приватний ключ – це рядок даних, що дозволяє отримати доступ до маркерів у певному гаманці. Вони діють як паролі, які приховані від будь-кого, крім власника адреси.

Блокчейн комісія – плата за транзакція або виклик смарт-контракту, яка нараховується користувачам при здійсненні транзакцій на блокчейні. Комісія збирається як винагорода валідаторам транзакції.

Confirmation – кількість підтверджень, які блок має, відноситься до того, скільки блоків було додано поверх цього блоку на блокчейн. Кожен доданий блок вважається підтвердженням, оскільки всі вузли в мережі опосередковано також перевіряють (підтверджують) блоки перед ним знову. Отже, якщо до блоку додано 5 блоків, цей блок має 5 підтверджень. Чим більше підтверджень має блок, тим менша ймовірність його зміни (важче атакувати), тому більш безпечною вважається транзакція.

Python – є інтерпретованою мовою програмування високого рівня загального призначення. Створений Гвідо ван Россумом і вперше випущений в 1991 році, філософія дизайну Python підкреслює читабельність коду з його помітним використанням значних пробілів. Його мовні конструкції та об'єктно-орієнтований підхід спрямовані на те, щоб допомогти програмістам написати чіткий, логічний код для малих і великих проектів.

Django – це основана на Python безкоштовна веб-платформа з відкритим вихідним кодом, яка слідує архітектурному зразку MTV. Він підтримується незалежною організацією Django Software Foundation (DSF), створеною як некомерційний. Основна мета Django – полегшити створення складних веб-сайтів, керованих базами даних. Структура підкреслює можливість багаторазового використання і підключення компонентів, менше коду, низьку зв'язок, швидкий розвиток, і принцип не повторювати

себе. Python використовується в усьому, навіть для файлів налаштувань і моделей даних. Django також надає додатковий адміністративний інтерфейс створення, читання, оновлення та видалення, який генерується динамічно через інтроспекцію і налаштовується за допомогою моделей адміністратора.

Solidity – відома як мова програмування на високому рівні за контрактом. Ця платформа має подібний синтаксис до мови сценаріїв JavaScript. *Solidity* як мова програмування зроблена для розширення віртуальної машини Ethereum. Solidity – це мова сценаріїв, яка виконує процес перевірки та виконання обмежень під час компіляції, а не під час виконання.

Tornado – масштабуемий асинхронний фреймворк, написаний на мові Python і призначений розробки бекенду, тобто серверної частини.

API – це множина визначених методів для взаємодії різних компонентів.

Об'єктно-реляційне відображення (ORM) – є методикою програмування, в якій дескриптор метаданих використовується для підключення об'єктного коду до реляційної бази даних. Об'єктний код написаний на мовах об'єктно-орієнтованого програмування (ООП), таких як Java або C #. ORM перетворює дані між типами систем, які не можуть співіснувати в реляційних базах даних і мовах ООП.

Docker – це сукупність взаємодіючих компонент програмного забезпечення, які використовують віртуалізацію на рівні операційної системи, щоб розвивати розробку та постачання програмного забезпечення в рамках стандартизованих пакетів програм.

InterPlanetary File System (IPFS) – це протокол і мережа, призначена для створення адресного, однорангового методу зберігання та спільного використання гіпермедіа в розподіленій файловій системі. Подібно до торрента, IPFS дозволяє користувачам не тільки отримувати, але й завантажувати дані до IPFS. На відміну від централізовано розташованого

сервера, IPFS побудований навколо децентралізованої системи користувачів-операторів, які утримують частину загальних даних, створюючи стійку систему для зберігання і читання користувацьких файлів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Ще в 18 столітті автори знайшли спосіб відправити свої роботи поштою собі. Як би це не було дивно, в епоху цифрових технологій цей метод використовувався до цих пір. Відправка листа собі може бути розглянута як класичний термін доказу прав власності. Марка поштового відділення, що вказує час, є доказом створення продукту в певний момент часу. Лист вважається доказом в суді і є лише у випадку спору.

Використовуючи блокчейн, не потрібно друкувати листа, йти на пошту і витрачати час, оплачувати пошту, відправляти конверт, а потім зберігати її з можливістю відкривати тільки один раз в суді.

Використовуючи блокчейн, інформація про авторські права є безпечною та легко перевіряється у випадку правових спорів.

Потреби підприємств щодо зберігання, продажу та захисту авторського права:

1. Проведення автоматичного збереження контентів.
2. Можливість підтвердження прав власності.
3. Розділення передачі прав доступу.
4. Виконання на всіх ОС та пристроях.

Аналогічним продуктом, що реалізує ці можливості, є запуск PatentBot. Недоліком цієї системи є відсутність можливості, що відповідає за продаж контенту.

1.1. Patent bot

Основна цінність бота полягає в тому, що клієнт виконує весь процес реєстрації в одному вікні. Оперативно і в той же час в 3-4 рази дешевше, ніж аналогічна послуга на ринку. У мобільній версії клієнт не залишає месенджера під час всього процесу реєстрації. Після натискання кнопки «Оплатити», внутрішній веб-переглядач перенаправляє запит клієнта на

сервер оплати і надає сторінку для заповнення платіжної інформації. Після заповнення даних та успішної оплати платіжна система надсилає клієнту інформацію про оплату. Однією з переваг роботи з ботом є те, що весь процес, включаючи оплату, відбувається виключно в месенджері, без необхідності переходу на інші посилання.

1.2. Nxt protocol

Nxt protocol – протокол, що працює поверх блокчейну bitcoin і дозволяє записувати у нього певні дані. Даний сервіс не надає зручного веб інтерфейсу та немає можливості передачі прав на контент, а також проведення аукціонів.

1.3. Порівняння

Основним недоліком юридичної реєстрація є те, що це займає багато часу. Patent bot не надає можливості передавати права, а також є дуже дорогим.

Nxt protocol є самим безпечним рішення, так як записує дані до самого стійкого блокчейну. Дешевший у порівнянні з юридичним підходом та Patent bot. Проте, не надає можливості проводити аукціони на передавати права на контент, так як у блокчейні біткоіна немає смарт-контрактів.

1.4. Аналіз вимог до можливостей продукту

В процесі аналізу вимог до системи було виділено такі основні вимоги:

1) Можливість авторизовуватися у системі не за допомогою логіну/паролю, я за за допомогою приватного ключа користувача.

2) Можливість завантажити контент у зашифрованому вигляді до блокчейну Ethereum.
3) Можливість передати права на контент через транзакцію на блокчейні.
4) Можливість виставити свій контент або профайл для продажу або аукціону.
5) Можливість переглянути історію продажів та покупок певної людини (адреси на блокчейні).
6) Можливість залишати відгуки про контент або продавця на блокчейні.

В процесі аналізу вимог до системи було виділено такі додаткові вимоги:

1) Використання декількох блокчейнів для зберігання контенту.
2) Реалізація програмного API для п.1-4.
3) Реалізація десктопної версії з можливістю підключення до блокчейн-демонів локально.

1.5. Висновки

Огляд існуючих рішень показав, що є такі типи web-сервісів для захисту авторського права:

1. Юридичні сервіси, що надають можливість зареєструвати авторські права та забезпечити їх подальший захист згідно законодавства.
2. Сервіси, що надають можливість записувати дані до блокчейну, але бракують можливості передачі прав на контент.

Отже, після детального аналізу існуючих рішень, доцільним є розробити сервіс, який буде надавати наступні можливості:

а) Швидкодія системи.

б) Невелика вартість завантаження контенту.

в) Можливість записувати дані до різних блокчейнів.

г) Можливість передачі прав безпосередньо або за допомогою аукціону.

2. АНАЛІЗ МОВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ WEB-САЙТІВ

2.1. Обгрунтування вибору web-додатку в якості типу програмного забезпечення

Основна реалізація системи являє собою смарт-контракт і усі дії зі строренням контенту, передачею прав можна робити за допомогою блокчейн-демона. Але такий підхів є не досить зручним, оскільки тоді потрібно запустати ПО, яке буде займати багато ресурсів.

Тому веб-додаток до розробленого смарт-контракту є дуже доречним, оскільки це дасть значне спрощення інтерфейсу.

Також можна створити десктопний додаток із можливістю підключення до віддаленого блокчейн-демона, але тут знов виграє веб-додаток, тому що він може бути запущений локально і забезпечувати кросплатформеність за допомогою технології Docker.

Таблиця 2.1

Критерій	Web-додаток	Десктопна програма
Кросплатформеність	Працює на усіх девайсах із web-браузером, також може бути розгорнутий локально	Необхідне портування програми для кожної конкретної ОС
Швидкодія	Забезпечує максимальну можливу швидкодію	Потребує від користувача значну кількість ресурсів
Вартість розробки	Дешевший ніж десктопні програми	Дорожча ніж web-додаток
Інсталяція	Не потребує інсталяції на стороні клієнта	Потребує встановлення
Інтерфейс користувача	Уніфікований інтерфейс	Привичний для користувача інтерфейс відповідної ОС

2.2. Порівняння блокчейнів EOS та Ethereum

EOS – це смарт-платформна платформа, створена компанією Block.one. Світ блокчейн вперше почув про EOS в 2017 році, і він швидко піднявся з чергового проекту blockchain, щоб конкурувати з гігантом blockchain, Ethereum. Вона не тільки виконує розумні контракти, але й створює повністю децентралізовані програми, які не відрізняються від звичайних рішень. Іншими словами, ідея EOS полягає в тому, щоб додатки виглядали так, як вони використовують стандартне рішення. EOS досі є найбільшою ICO всіх часів.

Хоча EOS має такі ж цілі, як Ethereum, спосіб, яким платформи хочуть досягти цих цілей, значно відрізняється. Одним з основних застережень є те, що EOS жертвує суворою децентралізацією для поліпшення масштабованості та зниження трансакційних витрат.

2.2.1. Масштабованість

З точки зору масштабованості, існує багато проблем, які мають вирішувати обидві платформи. Поки Ethereum здатна розмістити до п'ятнадцяти транзакцій в секунду, і це ще не досить ефективне, щоб конкурувати з такими платіжними системами, як EOS. Схема електронних платежів обробляє кілька тисяч транзакцій в секунду в пікові дні. Проте нещодавно були запропоновані зміни, які могли б збільшити кількість оброблених транзакцій до 100 000 за секунду. Оптимізація першого шару, наприклад, шардінг або процеси другого рівня, такі як плазма, може збільшити продуктивність платформи. Ці рішення значно збільшать кількість транзакцій, які платформа може обробити в певний час.

Тим часом EOS вже здатна обслуговувати до 10 000 транзакцій в секунду. У цьому випадку EOS має рішення, яке називається міжблочним зв'язком, яке створює іншу блок-ланцюг EOS, де можна пройти більше транзакцій. Ці блокчейн з'єднують один з одним, і немає межі для створення іншого блочного ланцюга в EOS.

2.2.2. Вартість транзакції

Витрати на транзакції є ще однією проблемою для впровадження blockchain. З Ethereum кожна операція коштує газу. Складність транзакцій і обсяг мережі впливають на ціну. Це велика перешкода з точки зору бізнесу, оскільки компанії та клієнти повинні прийняти певну втрату контролю. EOS працює по-різному – трансакційні витрати відсутні. Замість того, щоб платити за газ, користувачі орендують свої маркери для покриття пропускної спроможності для оплати транзакції. З EOS, однак, замість того, щоб витрачати ETH, можна компенсувати покриття маркерів, коли ми вирішимо, що ми більше не хочемо надавати нашу операцію, продаючи наші маркери. EOS блокчейн вкладає лексеми, які покривають пропускну здатність транзакції.

2.2.3. Механізм консенсусу

На додаток до масштабованості та вартості транзакцій, EOS також відрізняється від Ethereum своїм консенсусним механізмом. Ethereum є моделлю перевірки роботи, а EOS є делегованою моделлю долі. У Ethereum кожен вузол повинен вирішити криптографічну хеш-головоломку, щоб підтвердити транзакцію на блокчейн. Незважаючи на те, що Ethereum залишається децентралізованою, ця концепція «перевірки роботи» ускладнює масштабування.

Ethereum не тільки повільний, але й використовує багато обчислювальної потужності. Як наслідок, це призводить до величезних витрат на енергію. Ethereum планує перейти до механізму консенсусу щодо доказування ставок у найближчому майбутньому через протокол Casper. Після того, як розробники Ethereum вирішать проблеми безпеки з Casper, проблема масштабування Ethereum може поліпшитися.

З EOS, однак, двадцять один вузол відповідає за створення нових блоків. Однак це не є децентралізованою платформою і що вона взагалі відмовляється від духу технології blockchain. Тим не менш, це гарантує масштабованість і продуктивність мережі. Користувачам не потрібно чекати

кожного вузла, щоб підтвердити транзакцію. Замість цього користувачі повинні чекати п'ятнадцяти з двадцяти одного загального вузла для досягнення консенсусу. Це робить EOS дуже масштабованим і здатним виконувати експоненціально більше транзакцій в секунду.

2.2.4. Смарт контракти

Розробники Ethereum розробили контрактну мову платформи Solidity. Ця мова дозволяє платформі писати смарт-контракти. Інтелектуальні контракти EOS зазвичай знаходяться в C ++, але розробники можуть використовувати будь-яку мову програмування, яка має компілятор, який перетворює свій байт-код у веб-збірку. Для розробників blockchain для початківців, можна було б зручніше почати кар'єру зі смарт-контрактами в EOS.

2.2.5. Висновки

Незважаючи на те, що є EOS перемагає по всіх параметрах як швидкодія, ціна та масшабованість, він є централізованим, що робить його в’язливим до перезапису. Оскільки у даній роботі неможливість перезапису є критичним критерієм, то було обрано менш швидку, але більш надійну платформу Ethereum.

2.3. Аналіз платформи Django для розробки web-сервісів

Django – це високорівневий веб-фреймворк Python, що дозволяє швидко розвивати безпечні та підтримувані веб-сайти. Побудований досвідченими розробниками, Django піклується про більшу частину турбот про веб-розробку, тому ви можете зосередитися на написанні програми без необхідності винаходити колесо. Це вільний і відкритий код, має процвітаюче і активне співтовариство, велику документацію, і безліч варіантів для безкоштовної і платної підтримки.

Django слідує філософії "Батарейки включені" і забезпечує майже все, що розробники можуть захотіти зробити з коробки. Оскільки все, що вам

потрібно, є частиною одного продукту, все працює безперешкодно, слідує послідовним принципам дизайну і має велику й найновішу документацію.

Django може бути (і був) використаний для створення практично будь-якого типу веб-сайту – від систем управління контентом і вікі, до соціальних мереж і новинних сайтів. Він може працювати з будь-яким клієнтським фреймворком і може доставляти вміст практично в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML тощо).

В той час, як він надає вибір майже для будь-яких можливостей (наприклад, кілька популярних баз даних, двигуни шаблонів тощо), він також може бути розширений, щоб використовувати інші компоненти, якщо необхідно.

Django допомагає розробникам уникнути багатьох поширених помилок безпеки, надаючи структуру, розроблену для того, щоб робити правильні речі для автоматичного захисту веб-сайту. Наприклад, Django забезпечує безпечний спосіб керування обліковими записами користувачів і паролями, уникаючи типових помилок, наприклад, введення інформації про сеанс у файли cookie, де вони є вразливими (замість того, щоб файли cookie містили лише ключ, а фактичні дані зберігаються в базі даних) або безпосередньо зберігати паролі замість хеша пароля.

Хеш пароля – це значення фіксованої довжини, створене шляхом надсилання пароля через криптографічну хеш-функцію. Django може перевірити, чи правильно введений пароль, запустивши його через хеш-функцію і порівнявши вивід із збереженим хеш-значенням. Проте через односторонній характер функції, навіть якщо збережене значення хеш-компромісу порушено, зловмисникам важко розробити оригінальний пароль.

Django забезпечує захист від багатьох вразливостей за замовчуванням, включаючи ін'єкцію SQL, крос-сайт сценаріїв, підробку запитів між сайтами та clickjacking (докладніше про такі атаки див.

Django використовує архітектуру на основі розподілених компонентів (кожна частина архітектури не залежить від інших, і, отже, може бути замінена або змінена, якщо це необхідно). Наявність чіткого поділу між різними частинами означає, що він може масштабувати збільшений трафік, додаючи обладнання на будь-якому рівні: кешування серверів, сервери баз даних або сервери додатків. Деякі з найбільш завантажених сайтів успішно масштабували Django, щоб задовольнити їхні вимоги (наприклад, Instagram та Disqus, щоб назвати лише два).

Код Django написаний з використанням принципів дизайну та шаблонів, які заохочують створення коду, що підтримує та повторюється. Зокрема, він використовує принцип Don't Repeat Yourself (DRY), щоб не було непотрібного дублювання, зменшуючи кількість коду. Django також сприяє групуванню відповідних можливостей у багаторазові додатки і, на більш низькому рівні, групує пов'язаний код у модулі (уздовж лінії шаблону контролера перегляду моделі (MVC)).

Django написаний на Python, який працює на багатьох платформах. Це означає, що ви не прив'язані до якоїсь конкретної серверної платформи, і можете запускати свої програми на багатьох носіях Linux, Windows і Mac OS X. Крім того, Django добре підтримується багатьма веб-хостинг-провайдерами, які часто надають спеціальну інфраструктуру та документацію для розміщення сайтів Django.

2.4. Аргументація вибору блокчейну як технології для зберігання данних

Блокчейн фактично є базою даних, тому що це цифрова книга, яка зберігає інформацію в структурах даних, що називаються блоками. База даних також зберігає інформацію в структурах даних, званих таблицями. Однак, хоча блокчейн є базою даних, база даних не є блокчейн. Вони не є взаємозамінними в тому сенсі, що хоча вони і зберігають інформацію, вони відрізняються за дизайном.

2.4.1. База даних

Традиційна база даних – це структура даних, яка використовується для зберігання інформації. Це включає в себе дані, які можуть бути запитані, щоб зібрати розуміння для структурованої звітності, що використовується суб'єктами для підтримки бізнес-рішень, фінансових та управлінських рішень. Уряд також використовує бази даних для зберігання великих наборів даних, які масштабуються до мільйонів записів.

Проста база даних зберігається в елементах даних, які називаються таблицями. Таблиці містять поля, які визначають тип запису, в якому зберігаються дані, які називаються атрибутами. Кожне поле містить стовпці, які означають поле і рядки, які визначають запис, що зберігається в базі даних.

Базу даних можна змінювати, керувати та контролювати одним користувачем, який називається адміністратором. База даних завжди має користувача, який працює як адміністратор БД, і що користувач має повний контроль над базою даних. Цей користувач може створювати, видаляти, змінювати та змінювати будь-який запис, що зберігається в базі даних. Вони також можуть виконувати адміністрування в базі даних, як оптимізація продуктивності та управління розміром до більш керованих рівнів. Велика база даних має тенденцію до зниження продуктивності, тому адміністратори можуть запускати методи оптимізації для підвищення продуктивності.

Адміністратор може потім делегувати певні ролі іншим користувачам, що дозволяє їм адмініструвати або керувати базою даних. Наприклад, адміністратор може делегувати роль користувачеві, що дозволяє їм створювати нових користувачів для бази даних. Коли щось піде не так, адміністратор і їхні представники можуть відновити базу даних з резервної копії. У корпоративному світі такі проблеми поширені. Збій серверів, і єдиним способом відновлення даних є відновлення бази даних з резервної копії.

База даних також є рекурсивною, тобто ви можете повернутися до повторення завдання на конкретному записі, змінити або видалити його. Адміністратори часто видаляють старі записи в базі даних, які вже резервувалися в іншій базі даних або вважалися застарілою інформацією. Наприклад, якщо у вас є запис для "John Smith" у поточній базі даних, яку потрібно оновити до нової адреси. Існує вже резервна копія попередніх адрес "John Smith" в архівованій базі даних, тому запис може бути оновлений новою адресою в поточній базі даних.

2.4.2. Блокчейн

Блок-ланцюг зберігає інформацію в блоках рівномірного розміру. Кожен блок містить хешированную інформацію з попереднього блоку для забезпечення криптографічної безпеки. Хеш використовує SHA256, що є односторонньою хеш-функцією. Ця хешована інформація – це дані та цифровий підпис з попереднього блоку, а також хеші попередніх блоків, які йдуть до самого першого блоку, створеного в блокчейн, який називається початковий блок. Ця інформація проходить через хеш-функцію, яка вказує на адресу попереднього блоку. Структура даних blockchain є прикладом дерева Merkle, яке використовується як ефективний спосіб перевірки даних.

Блокчейн використовує розподілену мережу вузлів, яка є децентралізованою. Децентралізація означає, що всі вузли мережі зберігають копію блочного ланцюга. Вузли або зберігають повну копію (повні вузли) блокчейна або виконують гірничі операції, або вони можуть виконувати обидві. Немає адміністратора для перевірки блоку транзакцій.

Після того, як блок буде додано до блокчейна, інформація буде незмінною і прозорою для всіх. Транзакції Blockchain не є рекурсивними, тобто вони не можуть повторюватися після перевірки в блоці. Блокчейн є високотолерантним, оскільки, якщо один або більше вузлів упущені, завжди будуть доступні інші вузли, які запускатимуть блокчейн. Ще одна перевага децентралізації полягає в тому, що вона може бути неприйнятною і

ненадійною, що дозволяє людям, які не знають або не довіряють один одному, здійснювати операції. Що робить блокчейн, це забезпечити таку довіру через прозорість, записуючи транзакцію і надаючи криптографічно безпечний спосіб обміну цінності.

2.5. Вибір мов програмування

2.4.2. Мова для написання смарт-контракту

Початково існувало три мови високого рівня для розробки інтелектуальних контрактів:

1) Mutan, Golang-подібна мова. Вона була застаріла в березні 2015 року.
2) LLL, мова, подібна до Lisp. Ще підтримується в ядрі, але практично не використовується.
3) Serpent, мова, схожий на пітон. Проте його більше не рекомендується використовувати.

Пізніше Solidity була введена як 4-я мова. Solidity може дуже добре замінити всі інші існуючі мови високого рівня.

Solidity – це об'єктно-орієнтована мова високого рівня для впровадження інтелектуальних контрактів. Інтелектуальні контракти – це програми, які керують поведінкою рахунків у межах платформи Ethereum.

Solidity є статично типізованою, підтримує успадкування, бібліотеки та складні користувацькі типи серед інших функцій.

За допомогою Solidity можна створювати контракти різних видів використання.

2.4.2. Мова для написання серверної частини

PHP і Python є популярними мовами програмування високого рівня, які мають сильний фон з відкритим кодом, а також надають комплексну конструкторську документацію. Основна відмінність між PHP і Python полягає в тому, що PHP широко використовується для веб-розробки, тоді як

Python – це загальноприйнята мова програмування. PHP є мовою сценаріїв на стороні сервера, на відміну від цього Python – це об'єктно-орієнтована мова сценаріїв.

Розробка PHP була започаткована в 1994 році Rasmus Lerdorf. Раніше акронім, який використовувався для PHP, був персональною домашньою сторінкою, яка була замінена на Hypertext Preprocessor. Його документація доступна онлайн безкоштовно, оскільки вона випускається з точки зору ліцензії з відкритим вихідним кодом. У старті PHP не підтримує об'єктно-орієнтоване програмування, яке було додано в пізніших версіях.

Більшість розповсюджених систем управління контентом використовують PHP, такі як Media wiki, Drupal, Joomla, WordPress і так далі, які дозволяють створення сайту без особливих навичок програмування. Основною перевагою PHP є те, що він доступний на кожному провайдері хостингу. PHP розглядається як найбільш вкорінене середовище виконання на сервері в даний час. Це може забезпечити кращий рейтинг пошукової системи та доступність для хостинг-провайдерів.

Техніка вбудованого коду була розроблена PHP, так що код вбудований безпосередньо в документ змісту. Цей метод вбудовування коду був дуже ефективним для статичних і невеликих веб-сторінок. Пізніше вбудований код був замінений на файли шаблонів, оскільки веб-розробка та додатки стали більш складними.

Невідоме перетворення типів використовується в PHP, отже, це слабка система типів. Наприклад, ціле число і рядок можна порівняти за булевим виразом; це може створити плутанину і невизначеність. Існує ще один недолік використання вбудованих операцій бази даних MYSQL безпосередньо в коді, оскільки системи баз даних тісно пов'язані з PHP певними можливостями.

Раніше об'єктно-орієнтовані парадигми не були реалізовані в PHP, і це легко навчитися для новачків-кодерів. Його синтаксис близький до таких

мов, як C і Java. PHP є дуже надійною мовою, яка забезпечує сильну базу користувачів і її розподіл.

PHP здається звичною мовою, оскільки він походить від синтаксису на основі C. Остання версія PHP підтримує об'єктно-орієнтоване програмування, де коди і модулі, що складаються з функцій, інкапсулюються в об'єкт.

Ключові відмінності між PHP і Python для написання серверної частини, тобто бекенду:

1) Серед PHP і Python, PHP найпоширеніший і широко використовуваний.
2) PHP і Python, обидві мови читаються, але Python є більш ремонтоздатним, ніж PHP і складається з дуже небагато ключових слів.
3) PHP допускає погані практики програмування, які призводять до багатьох помилок, пов'язаних з безпекою, хоча може використовуватися безпечно. Навпаки, Python надає більше функцій безпеки, ніж PHP.
4) Python підтримує функціональне програмування, тоді як PHP не пропонує функціональних парадигм.
5) PHP не підтримує виключення належним чином; навпаки, у python існує належне положення для обробки виключень.
6) У python використовується оператор "yield" для функції генератора. З іншого боку, PHP не має положення для потоків (паралельне програмування).

Розвиток мови Python розпочав у 1991 році Гвідо ван Россум. Вона була розроблена як повноцінна мова загального призначення, на відміну від PHP, яка не обіцяє використовуватися як мова веб-скриптів. Мова має стандартний дефакт, який був реалізований фундацією python.

Python також має фон з відкритим кодом, подібний до PHP. Незважаючи на те, що він пропонує спільну веб-структуру, яка підвищує її

гнучкість, але вона потребує більшої кількості програмних зусиль, сервер Zope Application використовується в основному веб-фреймворком python. Перевага python полягає в обговоренні медійних рейтингів.

Мова Python не підкреслює розробку веб-додатків. Там використовується інший метод для веб-фреймворків, таких як CGI, WSGI (інтерфейс шлюзу веб-сервера), який може бути корисним для зміни середовища і шлюзу веб-програми, не впливаючи на вихідний код, що робить його портативним. Проте використовувати WSGI для початківців-програмістів досить складно.

Мова, подібна до PHP, мова Python була розроблена із застосуванням об'єктно-орієнтованої парадигми, незважаючи на це, вона також підтримує процедурне і функціональне програмування. Синтаксис python простий і легкий в освоєнні. Він має сильну систему типів і використовує явні методи.

Python є більш зручним для читання, ніж PHP, оскільки його команди нагадують слова, що використовуються в природній англійській мові. Він орієнтований на аспекти, де модулі розділяють реалізацію.

2.4.3. Мова та фреймворк для написання клієнтської частини

Ключовою відмінністю між AngularJS і ReactJS є те, що React є бібліотекою, а Angular – фреймворком.

Ключові відмінності такі:

1) Кутовий більш підходить для великих, структурованих додатків, тоді як React призначений скоріше для менших і більш гнучких додатків.
2) Реакція є більш гнучкою, ніж Angular і дозволяє розробнику легко додавати будь-які бібліотеки та додаткові компоненти JavaScript.
3) Кутовий фронт використовує шаблони на основі розширеної версії HTML, тоді як React використовує JSX, XML-подібну мову, побудовану поверх JavaScript.
4) Простіше створювати програми без будь-яких конфігураційних файлів за допомогою утиліти CLI "Створити реакцію програми".

5) Обидва React і Angular мають бібліотеки для розробки матеріалів. Однак бібліотека React є більш потужною, ніж бібліотека Angular.
6) І Angular, і React є компонентними. Тим не менш, компоненти Angular є класами TypeScript. З їх допомогою ви можете зв'язати структуру і реалізацію під одним об'єктом. У той же час, компоненти React є функціями JavaScript з обмеженою кількістю входів і одним виходом.
7) React підтримує односторонню зв'язок, коли дані передаються зверху вниз. Кут має одностороннє з'єднання і двосторонній інтерфейс. Він також має інтерполяцію, тобто здатність зв'язувати компонент всередині тексту між HTML-тегами і призначенням атрибутів.

2.6. Порівняння мікросервісної та монолітної архітектури

Недоліки монолітної архітектури:

1) Гнучкість:

   Монолітна архітектура не гнучка. Не можна використовувати різні технології. Технологічний стек вирішується на початку і слідує по всьому. Як тільки розвиток дозріває, іноді стає важко модернізувати версії технологічних стеків, не кажучи вже про поступову адаптацію нових технологій.
2) Надійність:

   Це не надійно. Якщо один компонент перестає працювати, весь додаток може перестати працювати.
3) Швидкість розвитку:

   Розробка дуже повільна в монолітній архітектурі. Новим членам команди важко зрозуміти і змінити код великого монолітного додатка. Якість коду з часом знижується. Зі збільшенням розміру кодової бази IDE перевантажується і стає повільнішим. Чим більше програма, тим більше часу потрібно для запуску. Всі ці фактори мають величезний вплив на продуктивність розробників.

4) Побудова складних додатків: Важко побудувати складну програму через обмеження технологій.
5) Масштабованість: Монолітні додатки важко збільшити, як тільки вони стають більшими. Ми можемо створити нові екземпляри моноліту і попросити балансувальник навантаження розподілити трафік на нові екземпляри, але монолітну архітектуру не можна масштабувати зі збільшенням навантаження. Кожна копія екземпляра програми буде мати доступ до всіх даних, що робить кешування менш ефективним і збільшує споживання пам'яті та трафік вводу-виводу. Крім того, різні компоненти програм мають різні вимоги до ресурсів: один може бути інтенсивним, а інший – інтенсивно. Що стосується монолітної архітектури, ми не можемо масштабувати кожен компонент самостійно.

Переваги мікросервісної архітектури:

1) Гнучкість:

   Мікросервісна архітектура досить гнучка. Різні мікросервіси можуть бути розроблені в різних технологіях. Оскільки мікросервіс менший, база кодів є значно меншою, тому оновити версії стека технологій не так вже й складно. Крім того, ми можемо поступово приймати нові технології без особливих труднощів.

2) Надійність:

   Мікросервісна архітектура може бути дуже надійною. Якщо одна компонента перестає працювати, то увесь додато продовжує роботу. Ми можемо вирішити проблему у відповідній мікросервісі і негайно розгорнути її.

3) Швидкість розробки:

   Розробка досить швидка в архітектурі мікросервісів. Оскільки обсяг коду значно менше для мікросервісу, новим членам команди не важко зрозуміти та змінити код. Вони стають продуктивними з самого початку. Якість коду підтримується добре. IDE набагато швидше.

Мікросервісу потрібно набагато менше часу для запуску. Всі ці фактори значно збільшують продуктивність розробників.

4) Побудова складних додатків:

   Завдяки архітектурі мікросервісу легко створювати складні програми. Якщо можливості програми аналізуються належним чином, ми можемо розбити її на незалежні компоненти, які можуть бути розгорнуті незалежно. Потім, навіть незалежні компоненти можуть бути додатково розбиті на невеликі незалежні завдання, які можуть бути розгорнуті самостійно як мікросервіс. Вирішення меж мікросервісу може бути досить складним завданням. Насправді це еволюційний процес, але як тільки ми вирішимо про мікросервіс, його легко розробити, оскільки в технології немає обмежень.

5) Масштабованість:

   Масштабованість є головною перевагою в архітектурі мікросервісу. Кожен мікросервіс можна масштабувати окремо. Оскільки індивідуальні мікросервіси значно менші за розмірами, кешування стає дуже ефективним.

3. ОПИСАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1. Структура програмних засобів

3.1.1. Загальна структура

Програмне забезпечення реалізовано у вигляді мікросервісної архітектури. Структурна схема системи зображена на рис. 3.1

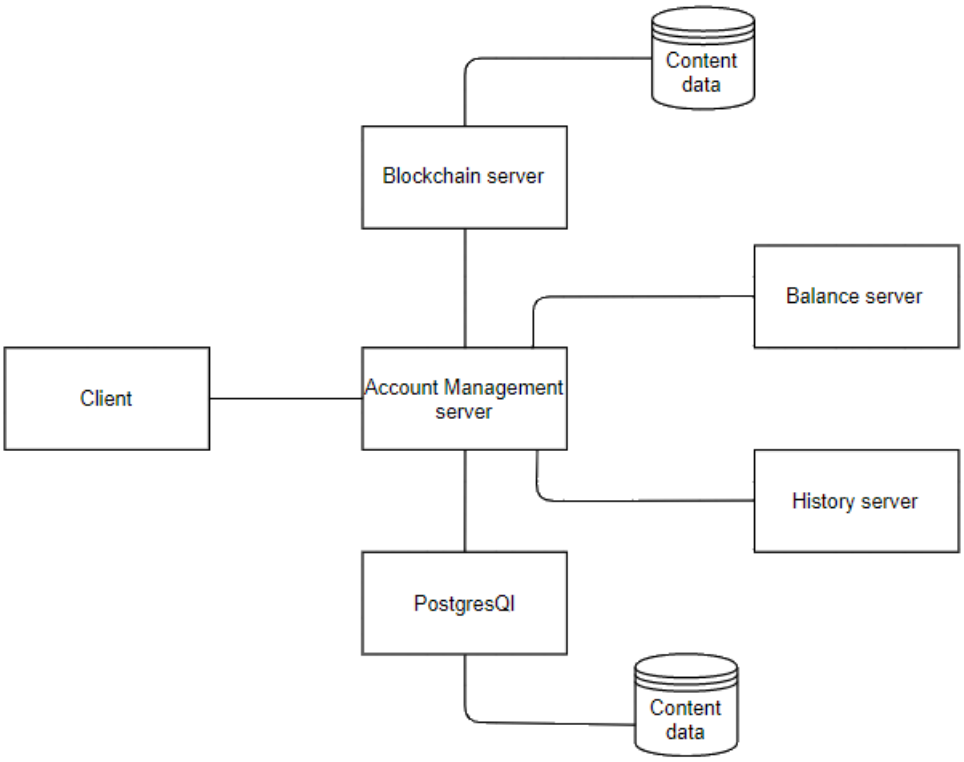


Рис. 3.1. Структурна схема системи

Account Management server обробляє запити від юзерів та виконує наступні можливості:

- Віддає користувачеві усі контенти, офрери та інші дані, що записані у блокчейні
- Забезпечує реалізацію депозиту та виводу із системи
- Обробляє логіку продажу контенту
- Формує відвовідні транзакції до блокчейну

Клієнт виконує взаємодію з користувачем, формує запити до сервера та показує результати роботи сервера.

3.1.2. Структура бази даних

На рис. 3.2 представлена ERD діаграма структури бази даних системи.

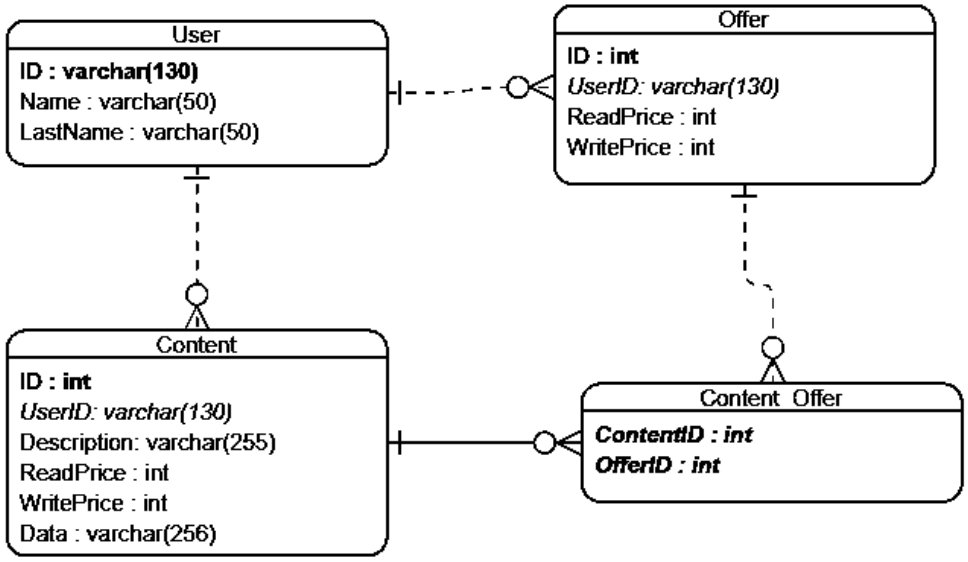


Рис. 3.2. Структура бази даних

Таблиця User – спеціальна таблиця стандартної підпрограми для авторизації платформи Django, що має всю основну інформацію про юзерів. Атрибути користувачів:

- Ім'я користувача;
- Прізвище користувача
- Публічний ключ для авторизації

Таблиця Content містить наступні поля

- Ціна для читання
- Ціна для перезапису або подальшого продажу
- Посилання на зашифровані дані на IPFS

Таблиця Content_Offer зв’язує офери із контентами.

Таблиця Offer містить наступні поля:

- Запропонована ціна для читання;

- Запропонована ціня для записк;
- ID користувача, який хоче купити контент користувача.

3.1.3. Описання взаємодії програмних компонентів

Blockchain сервер постійно моніторить транзакції у блокчейні із затримкою і якщо поповнюється адреса юзера, то передається запит на History server. Затримка потрібна для того, щоб транзакція отримала достатню кількість підтверджень. History server передаэ запит на Balance server, і якщо отримує Success від Balance Server, тоді записує в свою локальну базу даних з такими параметрами:

- Ідентифікатор користувача (uid)
- Ідентифікатор блокчейну (coinid)
- Кількість коінів (amount)
- Ідентифікатор транзакції (txid)

Для того щоб вивести токени із системи, від клієнтської частини приходить POST запит на AccountManagement серер із такими параметрама:

- Ідентифікатор користувача (uid)
- Ідентифікатор блокчейну (coinid)
- Кількість коінів (amount)
- Адреса, на яку потрібно вивести токени (address)

На MainExchange сервері створюється заявка і записується в базу даних unconfirmed_withdraw_database. Як тільки користувач авторизує запит, тобто підтвердить отримання листа на пошту, то заявка видаляється із бази данних на MainExchange сервері і передається на History сервер. Останній записує запит у свою базу данних і переадресовує на Balance сервер, який вже декрементує баланс користувача і відправляє запит на Blockchain сервер.

Для того щоб створити контент і записати його у блокчейн, з клієнтської частини потрібно відправити запит із такими параметрами:

- Ідентифікатор користувача (uid)
- Ідентифікатор блокчейну (coinid)
- Ціна для читання (read_price)
- Ціна для запису (write_price)

- Зашифрований контент (data)
- Описання контенту (description)
- Додаткова інформація (additional information)

Account Management сервер отримує запит і кешує усі параметри у локальній швидкій базі данних Clickhouse, а потім передає запит до Blockchain серверу, який записує контент до самого блокчейну.

Користувач може виконати запит до AccountManagement серверу на ендпоінт /search/products для того, щоб отримати список із усіма контентами. Також це можливо зробити самостійно через відповідний блокчейн-демон. Як тільки юзер вибрав контент, він може відправити запит на його покупку із такими параметрами:

- Ідентифікатор користувача (uid)
- Ідентифікатор блокчейну (coinid)
- Ціна для читання (read_price)
- Ціна для запису (write_price)
- Публічний ключ (public_key)

Публічний ключ потрібен для встановлення асиметричного каналу шифрування, щоб через нього передати пароль для контенту. AccountManagement сервер записує дані до своєї бази данних і передає запит до Blockchain серверу, який виконує запит на блокчейні.

Для того щоб подивитись усі активні офери, користувач повинен зробити запит на ендпоінт /offers. Щоб його обробити потрібно відправити запит до AccountManagement серверу дані із такими параметрами:

- Ідентифікатор контенту (cid)
- Ідентифікатор блокчейну (coinid)
- Статус (status)
- Публічний ключ (public_key)
- Зашифрованний пароль (password)

Статус може приймати значення «approve» та «reject». У першому випадку, поле password повенне містити пароль у зашифрованному вигляді.

3.2. Модель авторизації

Оскільки користувач може керувати своїм акаунтом лише за допомогою приватного ключа, може створювати і продавати контенти в обхід веб-сервісу, то звичайна авторизація у вигляді логіну та паролю не підходить. Тому у данному проекті застосована технологія цифрового підпису.

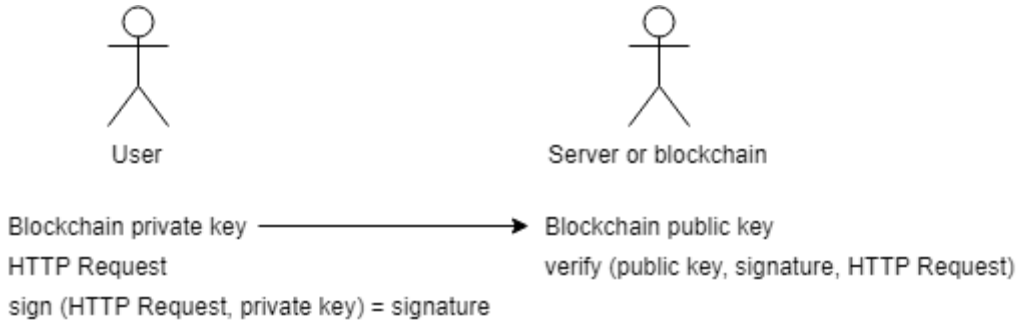


Рис. 3.3. Схема авторизації HTTP-запиту

Користувач володіє приватним ключем (Blockchain private key), з якого в односторонньому порядку можна отримати публічний ключ (Blockchain public key) – ідентифікатор на серверві або на блокчейні. Усі HTTP-запити користувач підписує своїм приватним ключем за допомогою функції sign, а сервер верифікує іх за допомогою функції verify, впевнюючись, що запит був підписаний приватним ключем, який відповідає публічному ключу користувача. Таким чином, відбувається авторизація HTTP-запиту. Для прискорення роботи системи у подальших запитах використувується технологія Secure cookie.

Модуль Client містить реалізацію для генерування приватного ключа у вигляді мнемоніки, а також файлу у якому записаний приватний ключ та пароль до нього.

Клієнт генерує використовує свій публічний ключ, який буде використовуватися для з’єднання. Він шифрує його за допомогою алгоритму, який також узгоджується під час фази Hello, і відкритого ключа сервера. Він надсилає цей зашифрований ключ на сервер, де він

розшифровується за допомогою приватного ключа сервера, на цьому частини рукостискання завершені. Сторони тепер можуть обмінюватись повідомленнями одне з одним і погодилися на ключ для симетричного шифрування даних, які вони збираються відправити один одному. HTTP-запити і відповіді тепер можуть бути надіслані шляхом формування відкритого текстового повідомлення, а потім шифрування і надсилання. Інша єдина, яка знає, як розшифрувати це повідомлення, і тому зловмисники Man In The Middle не можуть прочитати або змінити будь-які запити, які вони можуть перехопити.

3.3. Алгоритм передачі паролю

Асиметричне шифрування, яке часто називають шифруванням відкритого ключа, дозволяє продавцю контента відправити покупцю зашифроване пароль без спільного секретного ключа, так щоб сама система та інші користувачі не мали доступ до цього паролю.

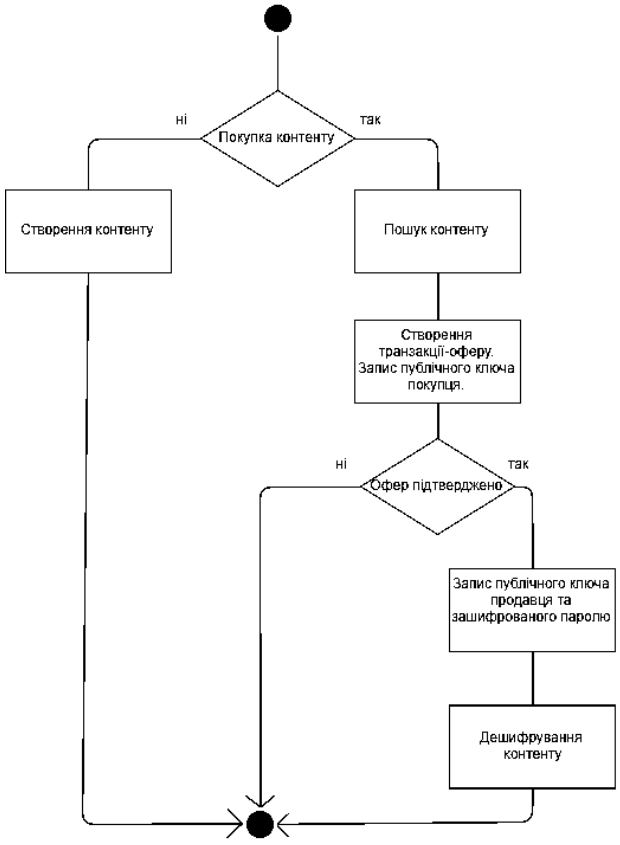


Рис. 3.4. Алгоритм передачі прав на контент

Покупець створює пару ключів, один з яких він зберігає і один з яких він посилає продавцю. Продавець шифрує пароль від контенту за допомогою ключа, який покупець послав до неї. Продавець посилає зашифровані дані покупцю. Покупець використовує свій секретний ключ, щоб розшифрувати дані і прочитати зашифрований пароль. Ключ, який Покупець посилає Продавцю – це відкритий ключ, і ключ, який він зберігає собі, – це "приватний" ключ; спільно вони формують "пару ключів" Покупця. Найбільш важливим аспектом асиметричного шифрування є те, що Продавець не бере участь у виборі ключа: Покупець створює пару ключів без будь-якого входу або згоди від Покупця і просто відправляє йому відкритий ключ. Покупець зберігає закритий ключ в секреті. Продавець використовує асиметричний алгоритм для шифрування повідомлення за допомогою відкритого ключа Покупця і відправляє йому зашифрований пароль, який він розшифровує за допомогою закритого ключа. Продавець не може розшифрувати зашифрований текст, який він створив за допомогою функції шифрування. Це означає, що Покупець може відправити свій відкритий ключ декільком людям, і кожен з них може створити зашифрований текст, який може розшифрувати лише секретний ключ Покупця. Односторонній характер функції шифрування означає, що повідомлення, створені одним відправником, не можуть бути прочитані іншим (тобто, Продавець не може розшифрувати зашифрований текст). Покупець може видавати публічний ключ кожному, хто хоче надіслати йому пароль, і він може навіть роздрукувати свій відкритий ключ на своїй візитній сторінці профайлу і роздати його всім, хто хоче відправити йому повідомлення.

Якщо Покупець підозрює, що інший користувач вгадав його приватний ключ, він просто створює нову пару ключів і розсилає новий відкритий ключ кожному, хто може надіслати йому повідомлення.

Основним обмеженням шифрування відкритим ключем є те, що він дуже повільний щодо симетричного шифрування і не є практичним для шифрування великих обсягів даних. Але оскільки за допомогою цього алгоритму потрібно зашифрувати невелику кількість данних, то цей алгоритм є вцілому дуже доречним.

3.4. Безпечність системи

Гаряча адреса – адреса, з якої токени переводяться на адреси користувачів. Приватні ключі гарячих адрес зберігаються у блокчейн-демонах у зашифрованому вигляді.

Холодна адреса – зовнішня адреса. Система не має приватних ключів до холодних адрес. Замість цього потрібно надіслати електронний лист із запитом на поповнення гарячої адреси.

Адреси користувачів – адреси, на які користувачі можуть вносити токени. Ці адреси генеруються за допомогою технології HD Wallet.

Зберігання і використання приватних ключів:

- Приватні ключі від адрес користувачів повинні бути імпортовані до відповідних демонів. Забезпечує реалізацію депозиту та виводу із системи.
- Усі приватні ключі зберігаються в демонах у зашифрованому вигляді wallet.dat
- Приватні ключі від холодних адрес повинні бути у адміністраторів системи.

Якщо гаряча адреса має занадто багато токенів, то блокчейн-демон надсилає надмірну кількість до холодної адреси. Якщо формула розраховує, що гаряча адреса має недостатньо токенів, демон надсилає електронного листа адміністрації. Формула має наступні параметри:

- Мінімальний баланс, який повинен бути на гарячій адресі. Блокчейн-демон завжди переводить токени на гарячю адресу так, що гаряча адреса має не менше ніж мігімальний баланс токенів.

– Мінімальний відсоток від усіх токенів у системі. Блокчейн-демон завжди перераховує токени на гарячю адресу так, що гаряча адреса має не менше відповідного мінімального відсотка коштів.
– Максимальний відсоток від усіх токенів у системі. Якщо гаряча адреса має більше токенів ніж відповідний максимальний відсоток, то блокчейн-демон переводить зайві токени на холодну адресу.

4. АНАЛІЗ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

4.1. Особливості реалізації

4.1.1. Серверна частина

На основі шаблону MTV Django були розроблено модуль AccountManagement для спрощення роботи із профайлами та користувацькими данними. Для реалізації серверу, який відповідає за роботиу із безпосередьо блокчейном було використано фреймворк Tornado, який дозволяє асинхронно і дуже швидко виконувати запити користувачів.

Оскільки виводи токенів із системи поступаюсь не одночас, а регулярно протягом певного часу і використання однііє транзакції на кожен вивід із системи було б недороченим, було створено спеціальний потоік WithdrawThread, який додає усі запити користувачів у базу данних, а потім виконує іх однією транзакцією.

4.2. Дизайн та вміст сторінок

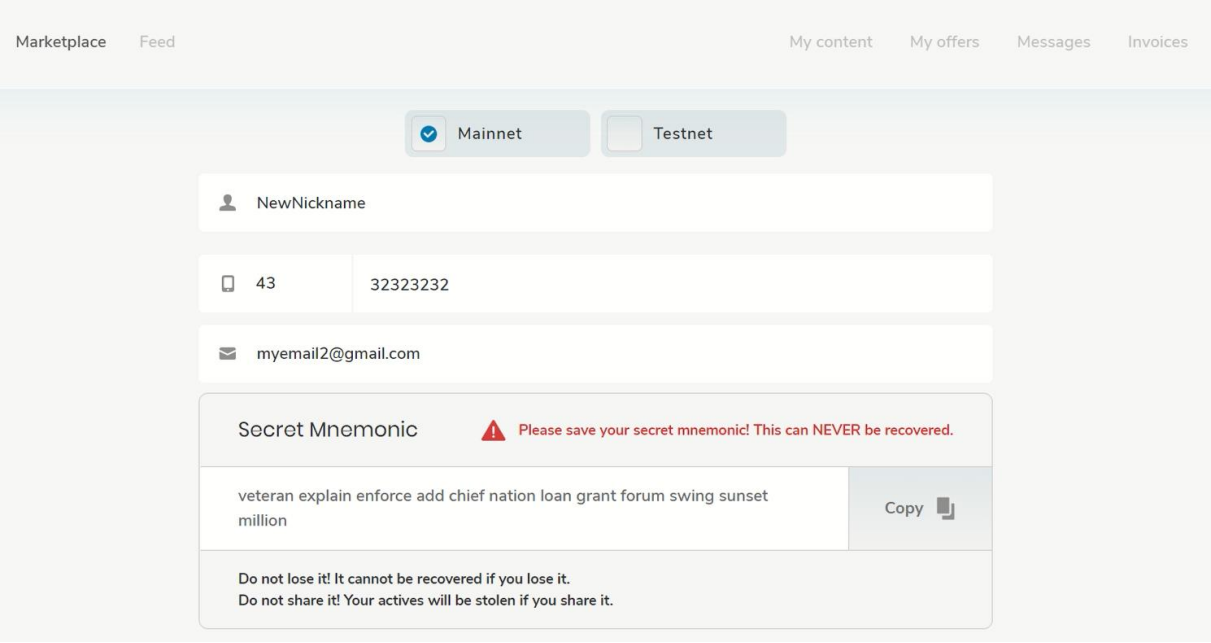


Рис. 4.1. Сторінка для реєстрації

Основною сторінкою з якої користувач починає свою роботу є сторінка. У верхній частині веб-додатку розміщені елементи головного меню, які стануть доступними одразу після авторизації.

У формі для реєстрації занходяться дві кнопки – Testnet, Mainnet. Вони означаються режим, у якому буде данна система використовуватись. Якщо обрано Testnet, то робота буде проходити на тестовому блокчейні, а якщо Mainnet, то у реальному блокчейні із реальними токенами.

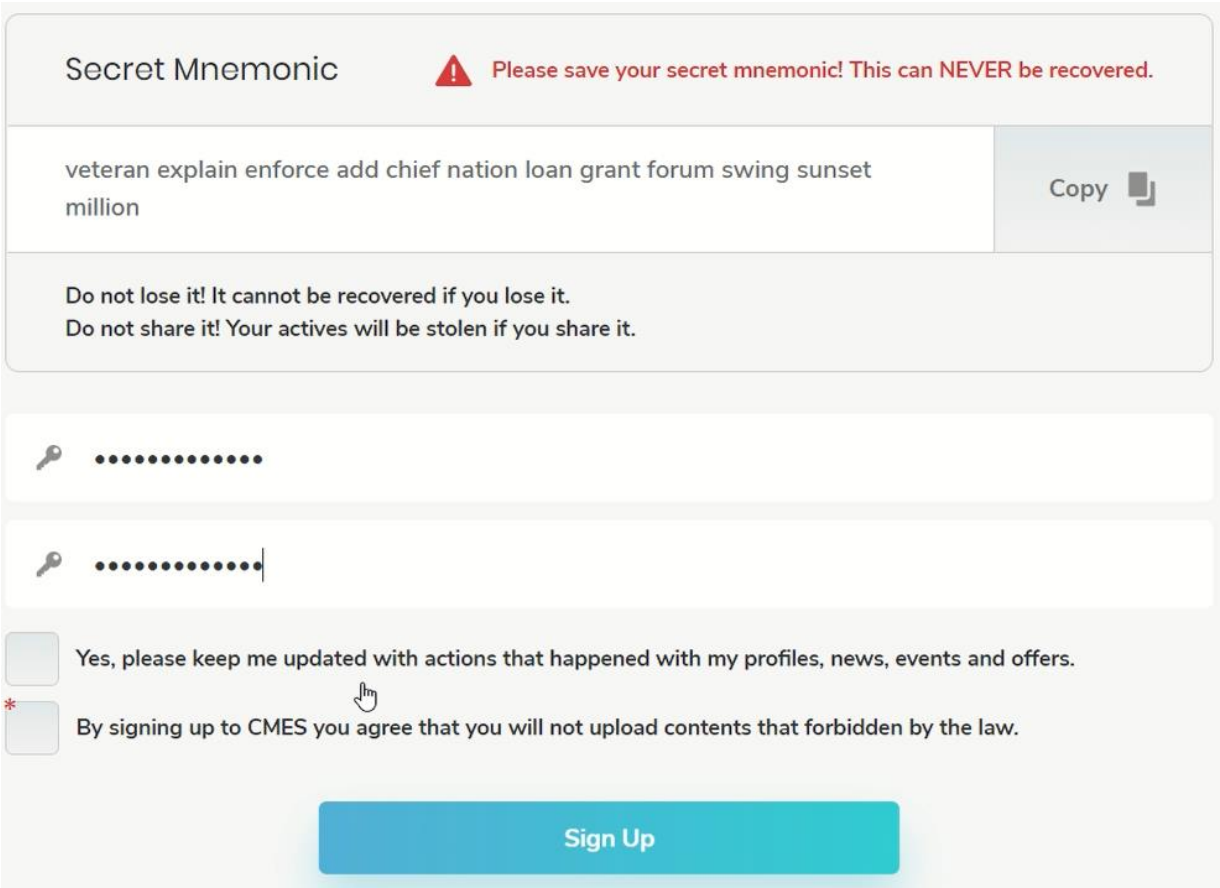


Рис. 4.2. Сторінка із блокчейн параметрами

Користувачеві генерується за домопогою клієнтського JS його блокчейн параметри:

– Мнемоніка – 12 слів, які представляють його приватний ключ, яким користувач повинен підписувати усі запити на сервер.
– Пароль – вводить сам користувач для шифрування файла із приватним ключем.

– Файл із зашифрованим приватним ключем – користувач повинен його зберігти.

Таким чином, як буде наведено далі, авторизація можлива у два способи: за допомогою мнемоніки і файла разом із паролем.

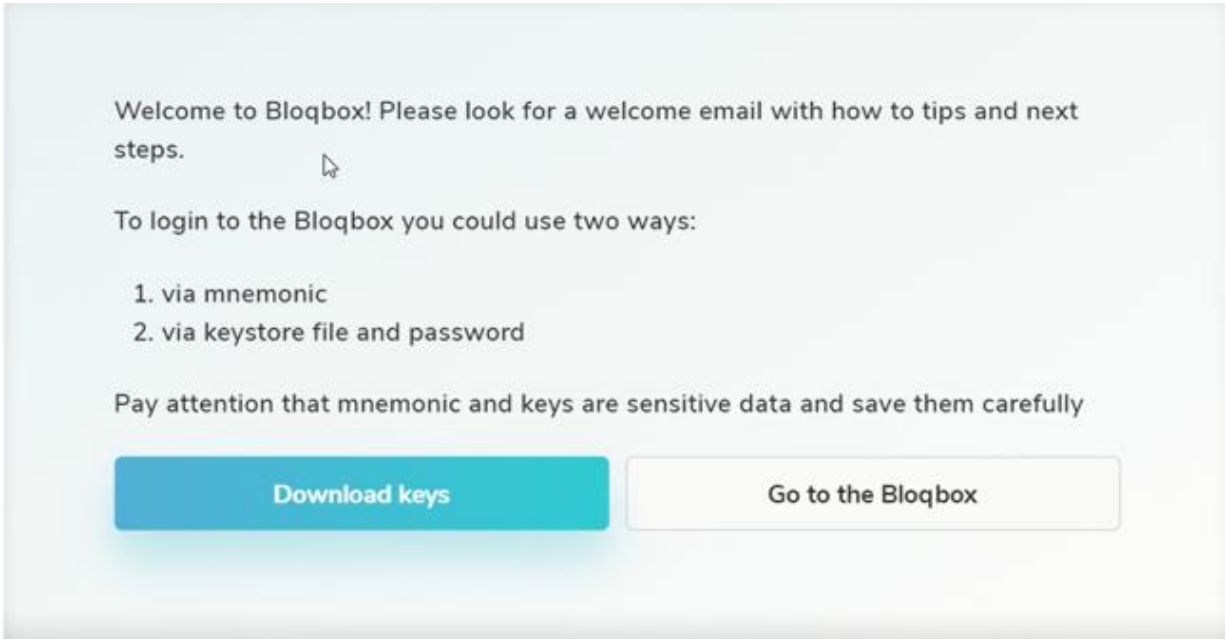


Рис. 4.3. Повідомлення про успішку реєстрацію

Після натискання на кнопку «Sign Up», користувачу виводиться повідомлення про успішну реєстрацію і пропонується зберегти файл із приватним ключем. Після натискання на кнопку «Download keys», веб-браезер зберігає данний файл на диску.

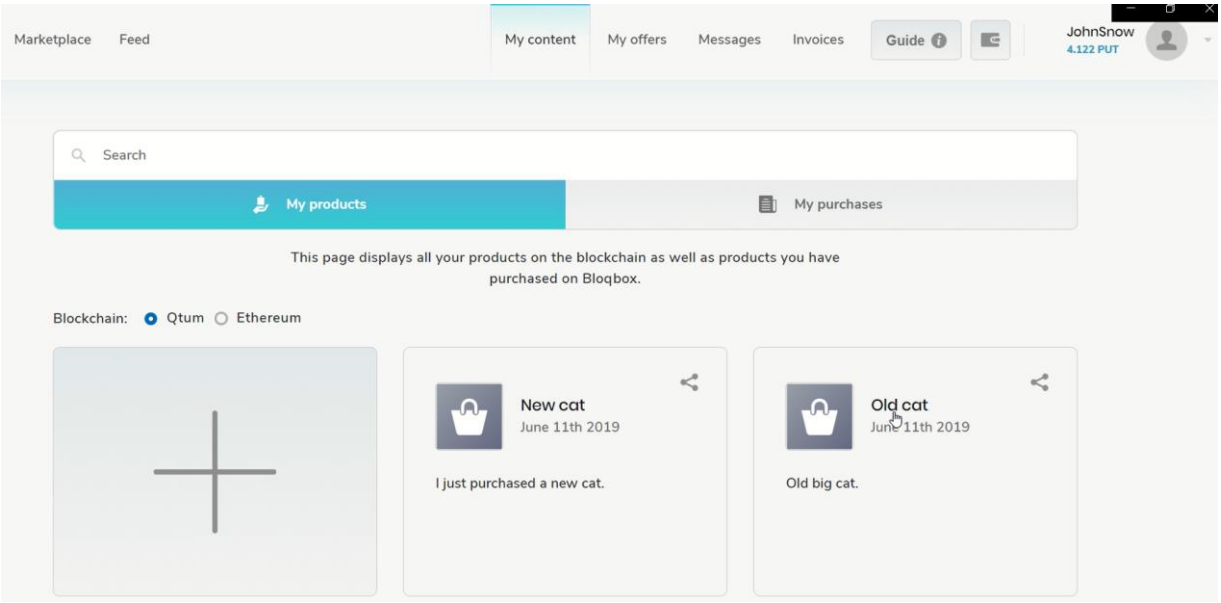


Рис. 4.4. Сторінка із контентами користувача

Після авторизації у системі, користувач поже переглянути свої контенти або придбані контени у вкладці «My content». Є поле для пошуку, а такжо кнопка-перемикач – «Blockchain», яке дозволяє преглядати контенти в залежності від блокчейну.

Перша кнопка у списку є посиланням на сторінку для створення контенту.

Доступною є також кнопка «My purchases», яка показує лише придбані контенти.

У правому верхному кутку показується аватар корустувача та його баланс у системі.

У вкладці «Marketplace» показується аналогічне меню, але з контентами усіх інших користувачів.

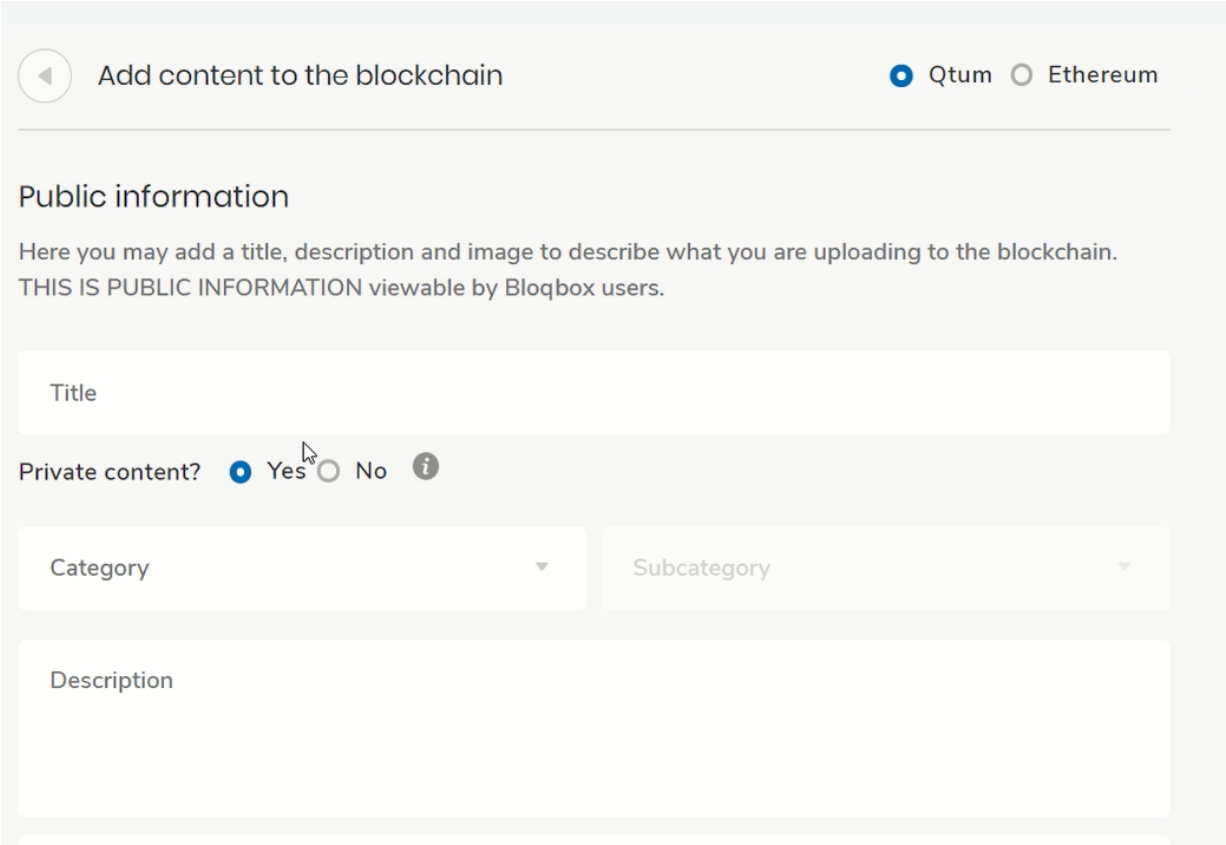


Рис. 4.5. Форма для заповнення контенту

Після натискання на кнопу «+» на (рис. 4.4), користувач може перейти до форми створення контенту з такими полями:

– Назва, яка буде показуватись на у заголовку контента.
– Кнопка-перемикач, яка означає чи буде контент доступний для продажу. Опцію «Yes» краще використовувати, якщо потрібно тільки для захисту авторських прав на контент. Опцію «No» краще використовувати, якщо контент передбачається для продажу.
– Категорія: музика, фотографії, програми тощо.
– Підкатегорія: чим точніше ви вкажете свою категорію, тим легше іншим користувачам буде знайти ваш контент.

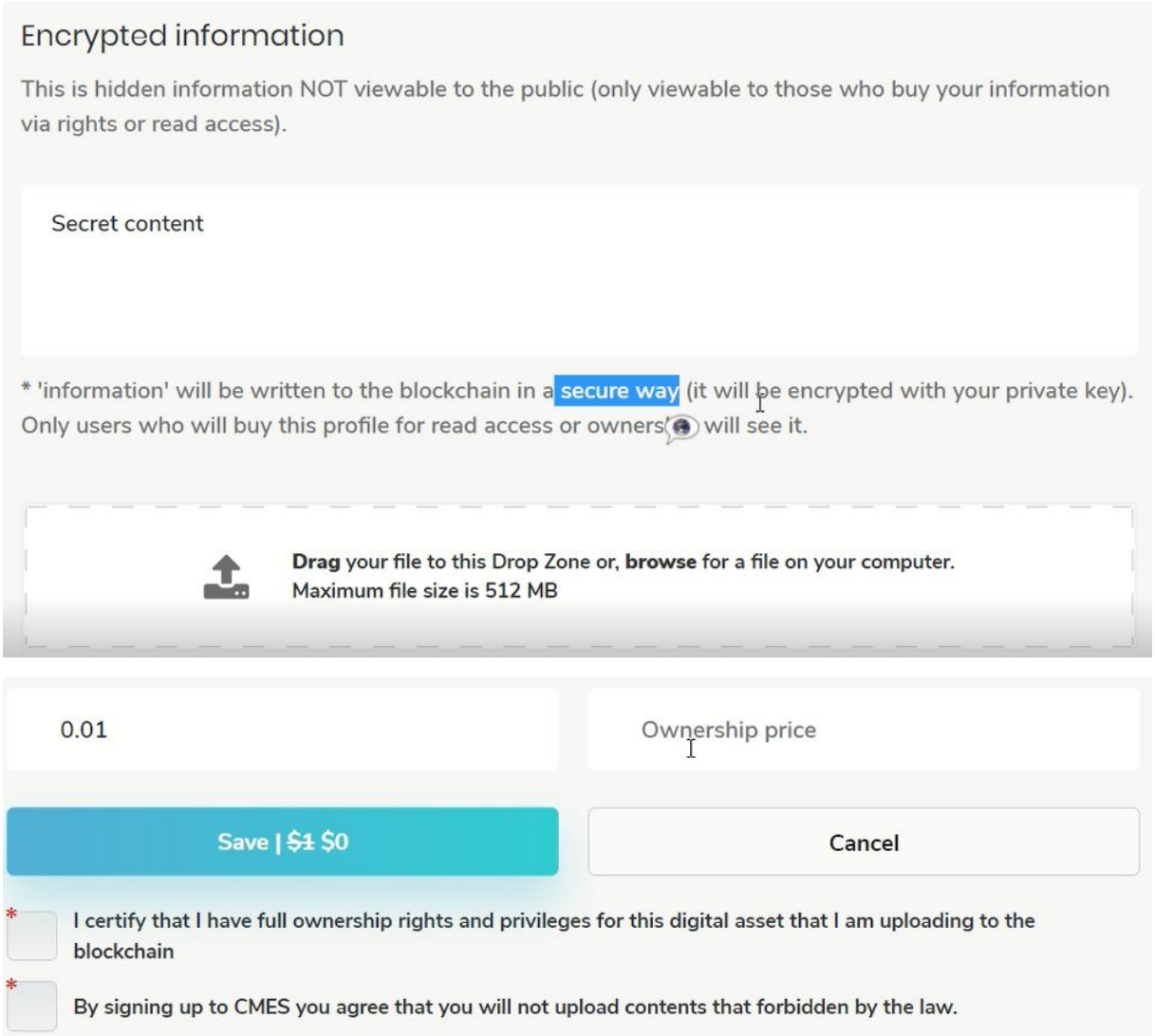


Рис. 4.6. Поле для вводу інформації про контент

Записати можна контент можна або у вигляді тексту або надати файл, який потрібно зашифрувати і записати в блокчейн. У другому полі потрібно перетягнути потрібний файл.

Можна виставити дві ціни:

– Ціна для читання – ціна, яку користувач повинен сплатити для того, щоб переглянит вміст файлу.
– Ціна для запису – ціна, яку користувач повинен сплатити для того, щоб мати право редагувати контент, змінювати самі ціни, а також продавати від свого імені.

Для того, щоб записати контент до блокчейну, користувач повинен погодитися із привилами системи:

– Права на контент належать йому.

– Вміст контенту не протирічить вимогам законодавства.

Після натискання кнопки «Save», контент записується на блокчейн і його можна побачити у вкладці «My contents».

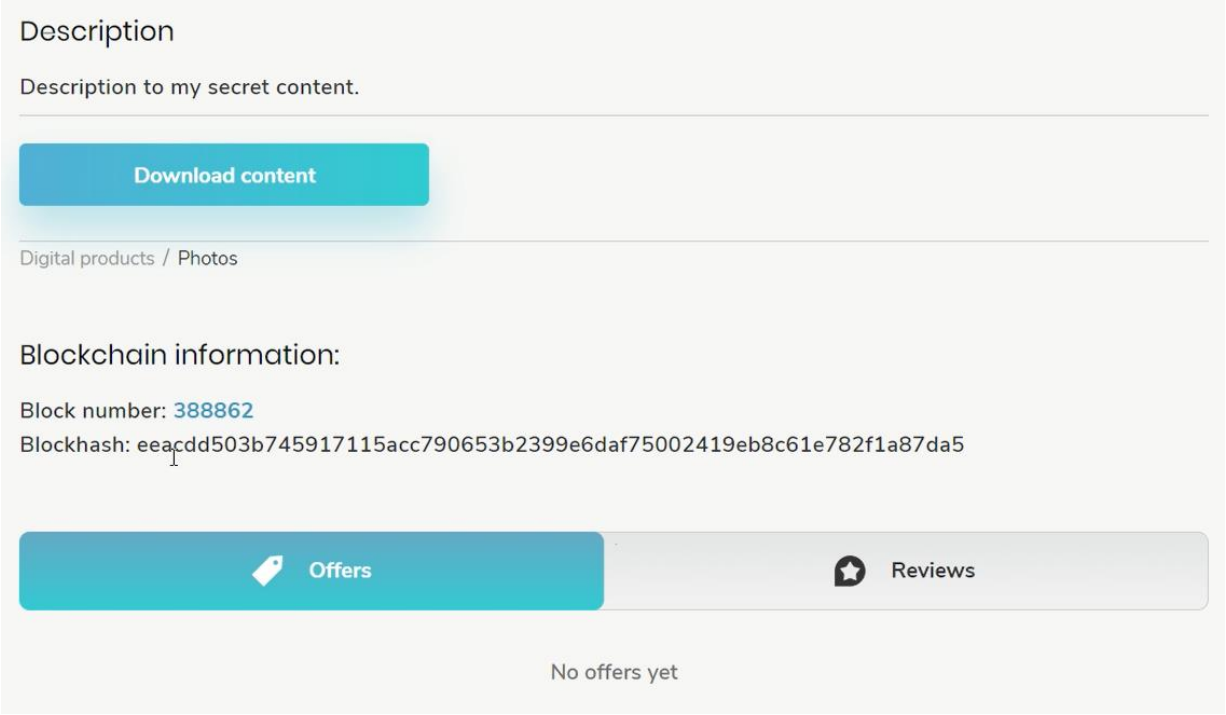


Рис. 4.7. Сторінка контенту

У сторінці контенту можна переглянути відповідну блокчейн інформацію, таку як номер та хеш блоку. Також показуються офери на покупку і відгуки.

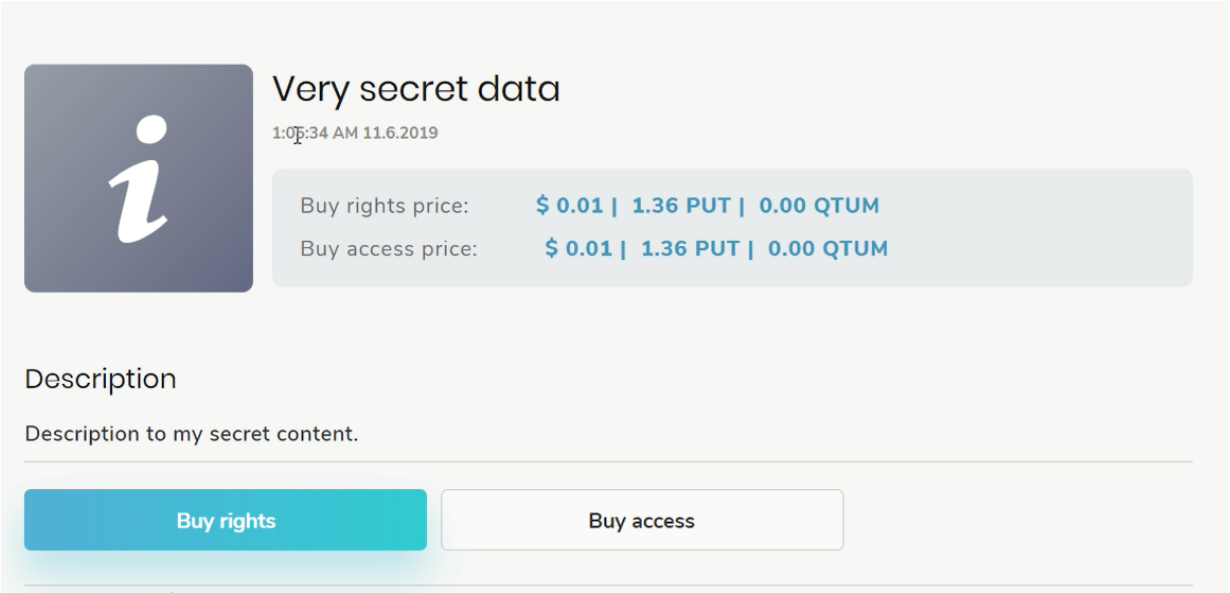


Рис. 4.8. Сторінка контенту для покупки

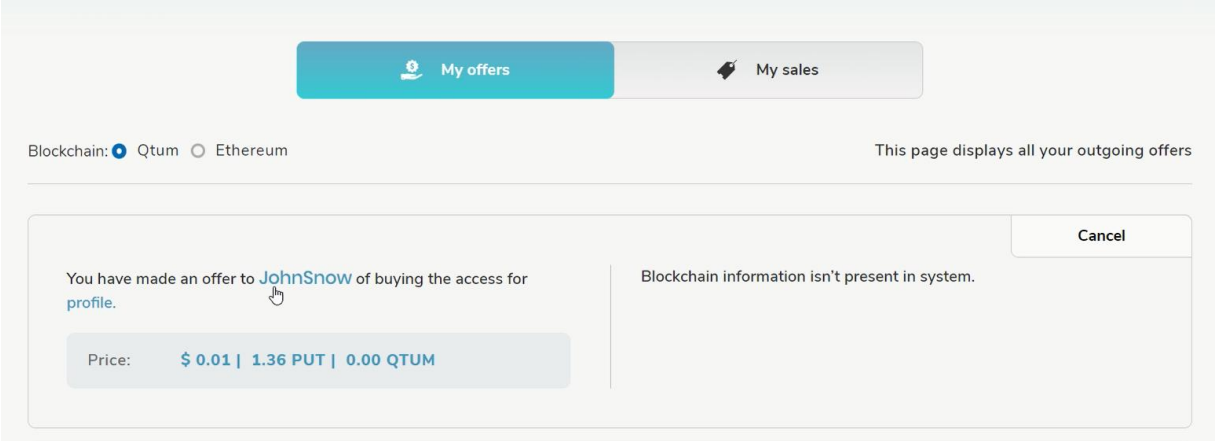


Рис. 4.9. Сторінка із оферами

Після того як було зроблено офер, покупець зможе переглянути його на вкладці «My offers». Справа, після того як транзакція підтвердиться, показується інформація про номер блоку та його хеш. Продавець у свою чергу зможе його побачити на вкладці «Invoices». Якщо покупець передумав покупати контент, він може натиснути на кнопку «Cancel», звісно це потрібно зробити до того як продавець підтвердить цей офер.

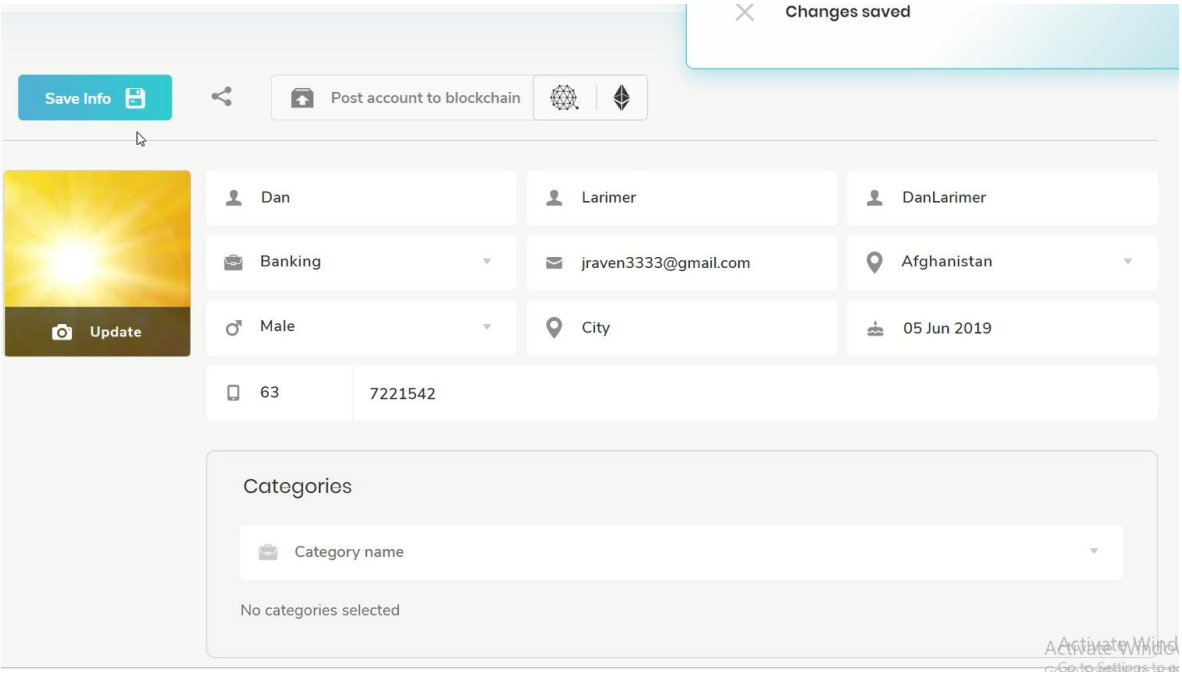


Рис. 4.10. Сторінка редагування профайлу

Після натиснення на кнопку «Feed», користувач має можливість змінити інформацію про свій профайл.

– Прізвище
– Ім’я
– Нікнейм
– Країна
– Поштова скринька
– Місто
– Дата народження

Також є можливість змінити аватар.

Свій баланс, я також адреси для поповнення можна дізнатися при натисканні на кнопку «Deposit».

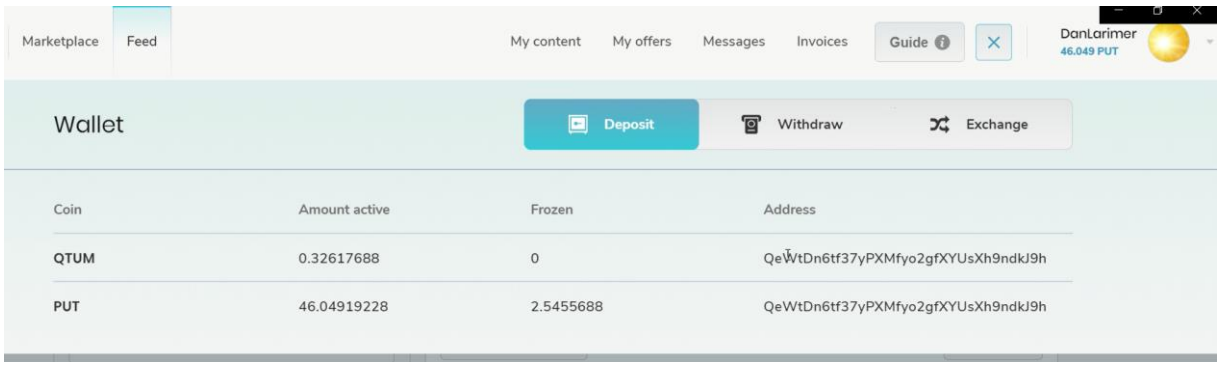


Рис. 4.11. Сторінка для поповнення токенів

Справа у колонці «Address» наведено адреси, на які потрібно переказати токени для того, щоб використовувати їх у системі.

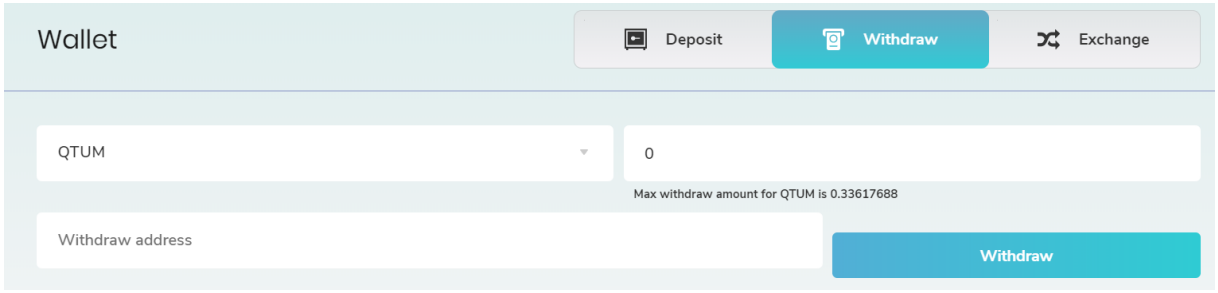


Рис. 4.12. Сторінка для виводу токенів із системи

Для того щоб вивести токени із системи, користувач повинен вказати кількість та тип токенів, а також заповнити поле із адресою. Якщо після заповнення форми, жодне з полів не підсвічується і нема повідомлень про помилки, тоді можна ініціювати процедуру виводу токенів із системи, натиснувши на синю кнопку «Withdraw» у правому нижньому кутку. Після цього користувач отримає ідентифікатор транзакції переводу токенів на блокчейні, перевірити яку можна, скориставшись такими ресурсами, яка «qtum info» або «qtum explorer». Після отримання достатньої кількості підтверджень, баланс користувача буде відмінусовано.

4.3. Тестування web-додатку

Для перевірки правильності роботи програмних засобів було розроблено набір тест-кейсів для проведення димового тестування.

Сценарій димового тестування:

1. Встановити систему.
2. Авторизуватися у системі за домомогою мнемоніки.
3. Поповнити адресу у системі.
4. Створити контент на блокчейні.
5. Авторизуватися у системі від імені іншого користувача.
6. Перевести токени до адреси у системі.
7. Знайти контент першого користувача.
8. Зробити офер першому користувачеві на покупку.
9. Авторизуватися у системі від імені першого користувача.
10. Підтвердити офер друго користувача.
11. Авторизуватися у системі від імені другого користувача.
12. Дешифрувати контент.

Таблиця 4.1

Тест-кейси димового тестування

№ п/п	ID вимоги	Опис	Очікувані результати
1	–	1. Встановити систему.	1. Система встановлена без помилок і доступна із web-браузерів користувачів.
2	1	1. Ввійти в систему. 2. Ввести мнемоніку першого користувача. 3. Підтвердити авторизацію.	1. Система показує сторінку авторизації. 2. Логін показуються коректно, мнемоніка показується у вигляді 12 слів. 3. Користувач переходить до сторінки із усіма контентами на блокчейні.
3	7	1. Перейти до вкладки «Deposit». 2. Відкрити додаток «Qtum web wallet».	1. Користувач переходить до сторінки для поповнення. 2. Користувач переходить до сторінки із можливістю переведення токенів до іншої адреси.

Продовження табл. 4.1

		3. Перевести токени до адреси у системі. 4. Перейти до вкладки «Deposits».	3. Користувач переходить до сторінки із усіма системними адресами. 4. Баланс користувача у системі збільшився.
4	5	1. Натиснути на кнопку «Create content». 2. Заповнити форму створення контенту. 3. Натиснути на кнопку «Save».	1. Користувач переходить до сторінки створення контенту. 2. Усі поля показуютсья коректно. Немає повідомлення про невірно введену назву. 3. Контент створений і користувач переходить до сторінки із усіма його контентами.
5	5	1. Натиснути на кнопку «Sign out». 2. Перейти до сторінки авторизації. 3. Вказати мнемоніку другого користувача. 4. Підтвердити авторизацію.	1. Користувач перестає бачити логін та баланс. 2. Користувач преходить до сторінки авторизації. 3. Поле із вводом мнемоніки відображається коректно, немає вспливаючого вікна із помилками. 4. Користувач переходить до сторінки із усіма контентами на блокчейні.
6	5	1. Перейти до сторінки із депозитами. 2. Відкрити додаток «Qtum web wallet». 3. Перевести токени до адреси у системі. 4. Перейти до сторінки із депозитами.	1. Користувач переходить до сторінки для поповнення. 2. Користувач переходить до сторінки із можливістю переведення токенів до іншої адреси. 3. Користувач переходить до сторінки із усіма системними адресами. 4. Баланс користувача у системі збільшився.
7	5	1. Натиснути на кнопку «Marketplace».	1. Користувач переходить до сторінки із усіма контентами на блокчейні.

		2. У полі для пошуку ввести назву контенту першого користувача.	2. Користувач переходить до сторінки із результатами пошуку.
8	1	1. Натиснути на ярлик контенту. 2. Натиснути на кнопку «Make offer». 3. Підтвердити офер.	1. Користувач переходить до сторінки із інформацією про контент. 2. Показується діалогове вікно для підтвердження дій користувача. 3. Офер записується на блокчейн. Баланс користувача зменшується на величину, яка дорівнює сумі ціни контенту та комісії.
9	2	1. Натиснути на кнопку «Sign out». 2. Перейти до сторінки авторизації. 3. Вказати мнемоніку першого користувача. 4. Підтвердити авторизацію.	1. Користувач перестає бачити логін та баланс. 2. Користувач преходить до сторінки авторизації. 3. Поле із вводом мнемоніки відображається коректно, немає вспливаючого вікна із помилками. 4. Користувач переходить до сторінки із усіма контентами на блокчейні.
10	3	1. Натиснути на вкладку «Invoices». 2. Натиснути на офер першого користувача. 3. Натиснути кнопку «Approve».	1. Користувач переходить до сторінки із усіма оферами. 2. Користувач переходить до сторінки із інформацією про офер. 3. Користувач записує пароль від контенту на блокчейн у зашифрованому вигляді. Після підтвердження транзакції, баланс збільшеється на величину, яка дорівнює сумі контенту.
11	4	1. Натиснути на кнопку «Sign out».	1. Користувач перестає бачити логін та баланс.

Продовження табл. 4.1

		2. Перейти до сторінки авторизації. 3. Вказати мнемоніку другого користувача. 4. Підтвердити авторизацію.	2. Користувач преходить до сторінки авторизації. 3. Поле із вводом мнемоніки відображається коректно, немає вспливаючого вікна із помилками. 4. Користувач переходить до сторінки із усіма контентами на блокчейні.
12	1	1. Натиснути на кнопку «My products». 2. Натиснути на придбаний контент. 3. Натиснути на кнопку «Download content».	1. Користувач переходить до сторінки із його контентами. 2. Користувач переходить до сторінки із інформацією про контент. 3. Вміст контенту показується на сайті і завантажується у незашфрованному вигляді на комп’ютер користувача.

4.4. Рекомендації щодо подальшого вдосконалення

Основною рекомендацією для вдосконалення додатку є додавання підтримки більшої кількості блокчейнів.

Також додавання нейронної мережі для ствоення гнучкої системи класифікації контентів.

Необхідно додати підтримку апаратного входу, тобто запис приватного ключа до таких пристроїв як Trezor або Lendger Nano S.

Також важливою можливістю являється можливість пошуку подібних контентів. Цю можливість рекомендується використати для оновленого пошуку та для знаходження подібних контентів на сторінці перегляду контенту.

Для срощення комунікації між продавцем та покупцем буде доречним створення чату онлайн, якщо їх ціни на контент розбігаються.

ВИСНОВКИ

Метою даного дипломного проекту було розроблення web-додатку для захисту авторського права і його передачі за допомогою технології блокчейн.

Аналіз засобів розроблення web-додатку, попередньо виконаний в дипломному проекті, показав доцільність створення системи у вигляді web-сервісу.

Розробленна автоматизована система:

- забезпечує авторизацію доступу за допомогою приватного ключа;
- дозволяє записувати контент у зашифрованому вигляді до блокчейну;
- дає можливість покупати та продавати контент;
- має зрозумілий та зручний інтерфейс користувача.

Особливу увагу під час розроблення даного програмного продукту було приділено безпеці, тобто нетривіальна авторизація за допомогою приватного ключа, а також реалізація безпечного алгоритму для передачі прав на контент.

Розробка виконана у повному обсязі, всі вимоги враховані, тестування продукту виконано у відповідності до затвердження програми та методики тестування.

Використання розробленої системи дозволить захист авторського права на підприємствах.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Blockchain – Wikipedia [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA%D1%87%D0%B5%D0%B9%D0%BD
2. Ethereum – Wikipedia [Електронний ресурс]. — Режим доступу: http://www.ethdocs.org/en/latest/
3. Assymetric encryption – Wikipedia [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Public-key_cryptography
4. HTTPS – Wikipedia [Електронний ресурс]. — Режим доступу: https://love2dev.com/blog/how-https-works/
5. Python – Вікіпедія [Електронний ресурс]. — Режим доступу: http://ru.wikipedia.org/wiki/Python
6. Python Documentation Index [Електронний ресурс]. — Режим доступу: http://www.python.org/doc/
7. PHP – Википедия [Електронний ресурс]. — Режим доступу: http://ru.wikipedia.org/wiki/PHP
8. Документація EOS [Електронний ресурс]. — Режим доступу: https://eos.io/
9. Документація Tornado [Електронний ресурс]. — Режим доступу: https://www.tornadoweb.org/en/stable/
10. Django на русском [Електронний ресурс]. — Режим доступу: http://djbook.ru/
11. Newman, S. Django 1.0 Template Development [Текст] / S. Newman. — Apress, 2008. — 252 с.
12. Robbins, J. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics [Текст] / J. Robbins. — O'Reilly Media, 2012. — 587 с.

13. HTML Tutorial [Електронний ресурс]. — Режим доступу: http://www.w3schools.com/html/
14. Newman, S. Django 1.0 Template Development [Текст] / S. Newman. — Apress, 2008. — 252 с.
15. Robbins, J. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics [Текст] / J. Robbins. — O'Reilly Media, 2012. — 587 с.
16. HTML Tutorial [Електронний ресурс]. — Режим доступу: http://www.w3schools.com/html/
17. Vue js [Електронний ресурс]. — Режим доступу : https://vuejs.org/v2/guide/
18. Онлайн документація Twitter Bootstrap [Електронний ресурс]. — Режим доступу : http://twitter.github.io/bootstrap/index.html

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКОГО ПРАВА НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.О. Саприкін

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмні засоби зберігання, продажу та захисту авторського права блокчейну являють собою web-сайт, написаний на мові програмування Python з використанням фреймворку Django та Tornado, а також фреймворків EthereumPy та IPFS.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Працездатність елементів сторінок web-сайту.
2. Наявність доступу до блокчейн-бази даних контентної документації.
3. Взаємодію сервера із блокчейном та ефективність роботи серверу.
4. Забезпечення коректної обробки запитів від користувача.
5. Забезпечення належного рівня безпеки даних.
6. Зручність роботи з web-сайтом.
7. Відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні Critical path test (базове тестування).
2. Тестування продуктивності програмного забезпечення, зокрема Performance testing (тестування стабільності).
3. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування web-ресурсу в різних web-браузерах.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування зручності використання.
8. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКОГО ПРАВА НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В.Онай

Виконавець:

_____ А.О. Саприкін

2019

ЗМІСТ

1. Описання структури програмних засобів

Програмні засоби включають в себе web-додаток, що складається із статичних web-сторінок та web-сторінок, вміст яких формується динамічно. Web-додаток є одномовним з англійською мовою.

До статичних належать наступні web-сторінки:

- Сторінка авторизації
- Сторінка перегляду вмісту контента для автора контента
- Сторінка редагування контента

Динамічна частина web-додатку включає наступні web-сторінки:

- Сторінка пошуку контентів
- Сторінка покупки контенту
- Сторінка профайлу
- Сторінка із балансами

Кожна web-сторінка містить інформацію про користувача, що ввійшов в систему, його роль та кнопку виходу із системи, а також логотип системи.

2. Описання реєстрації та авторизації

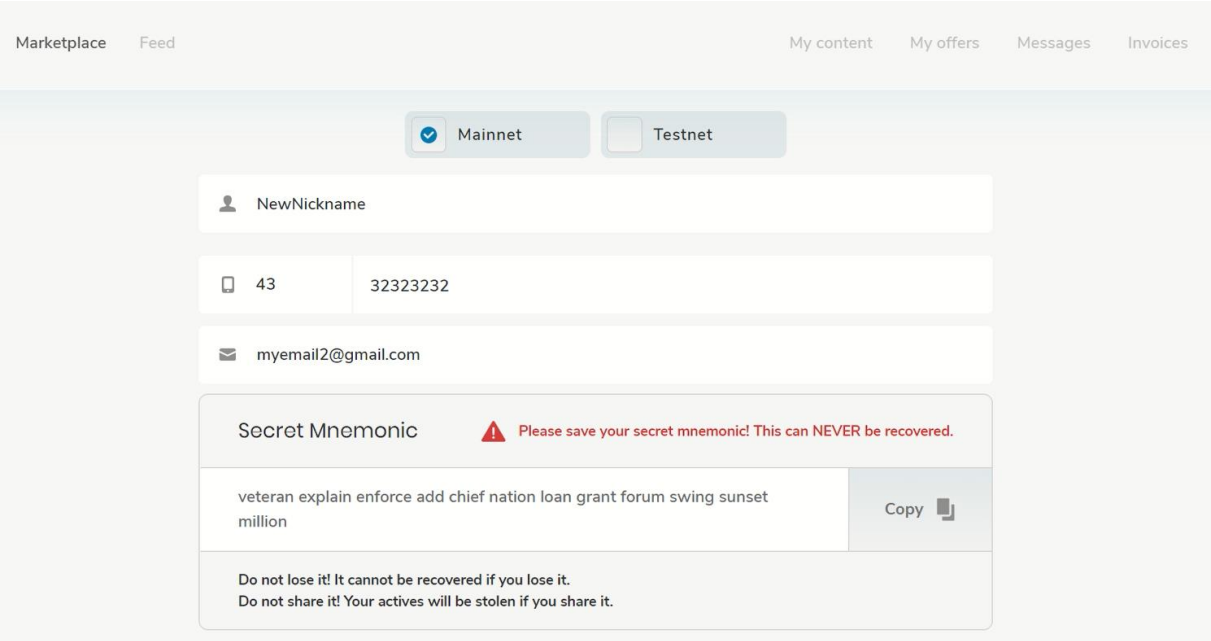


Рис. 2.1. Сторінка для реєстрації

Основною сторінкою з якої користувач починає свою роботу є сторінка. У верхній частині веб-додатку розміщені елементи головного меню, які стануть доступними одразу після авторизації.

У формі для реєстрації занходяться дві кнопки – Testnet, Mainnet. Вони означаються режим, у якому буде данна система використовуватись. Якщо обрано Testnet, то робота буде проходити на тестовому блокчейні, а якщо Mainnet, то у реальному блокчейні із реальними токенами.

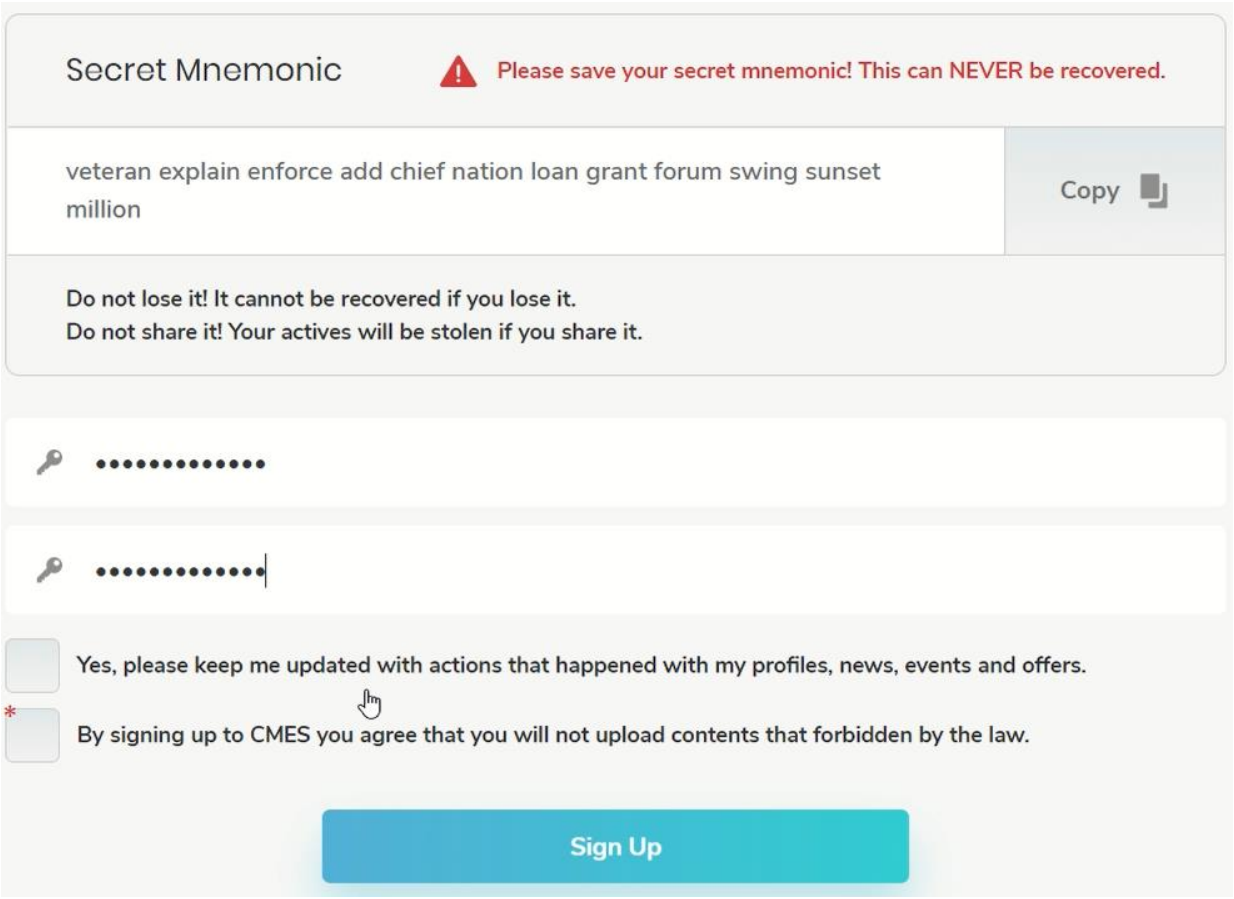


Рис. 2.2. Сторінка із блокчейн параметрами

Користувачеві генерується за домопогою клієнтського JS його блокчейн параметри:

– Мнемоніка – 12 слів, які представляють його приватний ключ, яким користувач повинен підписувати усі запити на сервер.
– Пароль – вводить сам користувач для шифрування файла із приватним ключем.

– Файл із зашифрованим приватним ключем – користувач повинен його зберігти.

Таким чином, як буде наведено далі, авторизація можлива у два способи: за допомогою мнемоніки і файла разом із паролем.

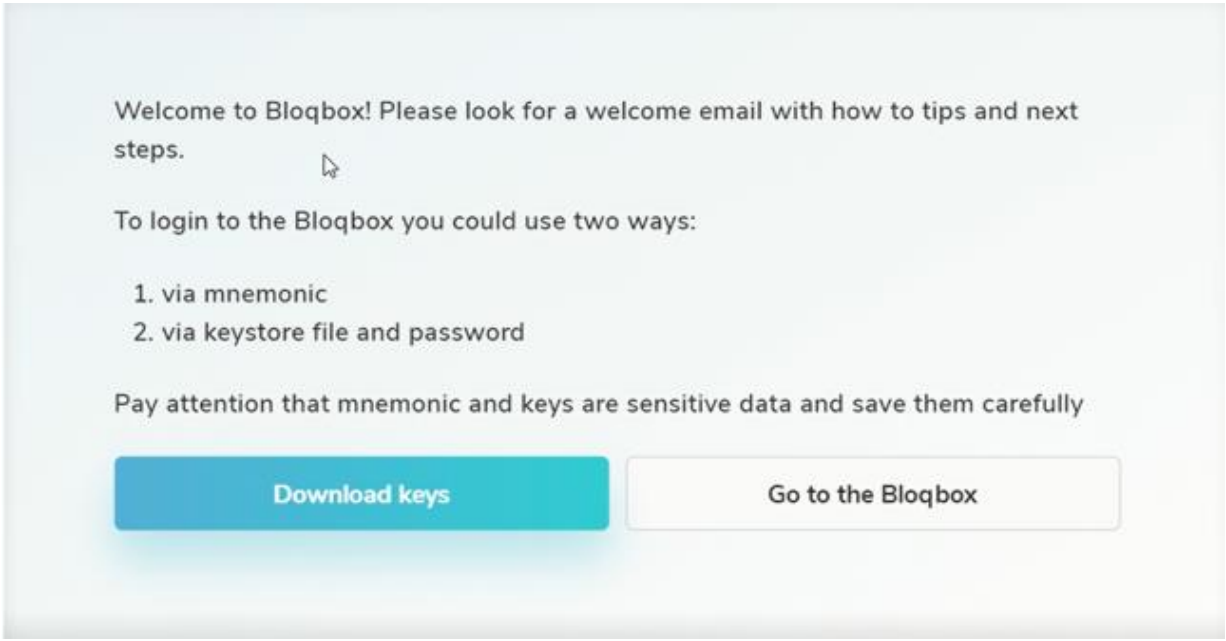


Рис. 2.3. Повідомлення про успішку реєстрацію

Після натискання на кнопку «Sign Up», користувачу виводиться повідомлення про успішну реєстрацію і пропонується зберегти файл із приватним ключем. Після натискання на кнопку «Download keys», веб-браезер зберігає данний файл на диску.

3. Менеджмент контентів

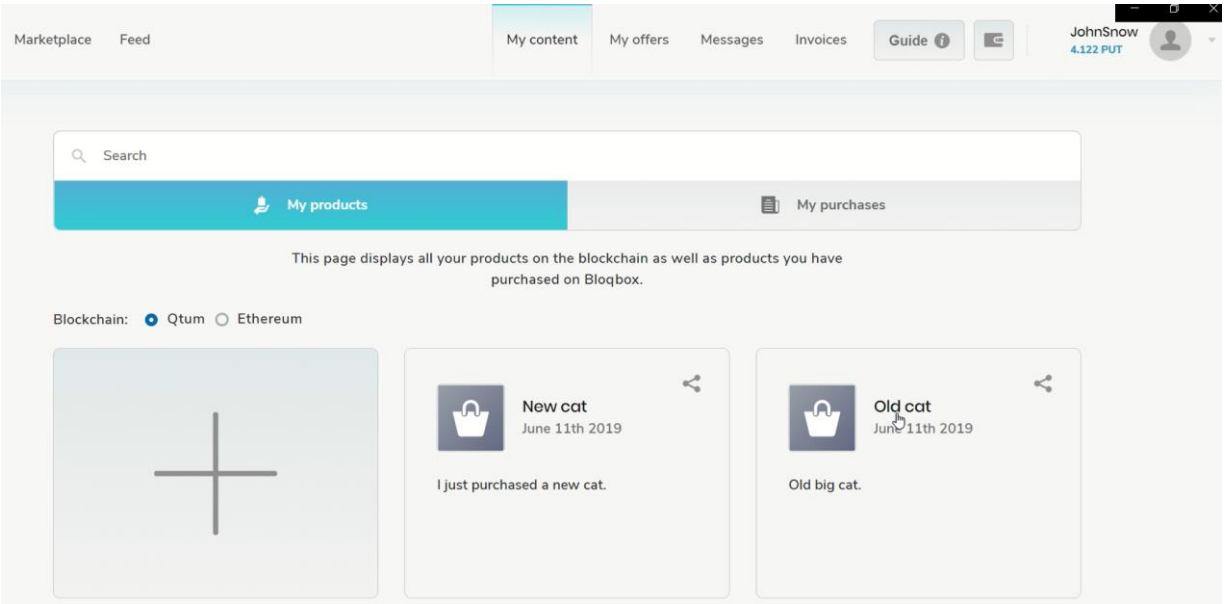


Рис. 3.1. Сторінка із контентами користувача

Після авторизації у системі, користувач поже переглянути свої контенти або придбані контени у вкладці «My content». Є поле для пошуку, а такжо кнопка-перемикач – «Blockchain», яке дозволяє преглядати контенти в залежності від блокчейну.

Перша кнопка у списку є посиланням на сторінку для створення контенту.

Доступною є також кнопка «My purchases», яка показує лише придбані контенти.

У правому верхному кутку відображається аватар корустувача та його баланс у системі.

У вкладці «Marketplace» відображається аналогічне меню, але з контентами усіх інших користувачів.

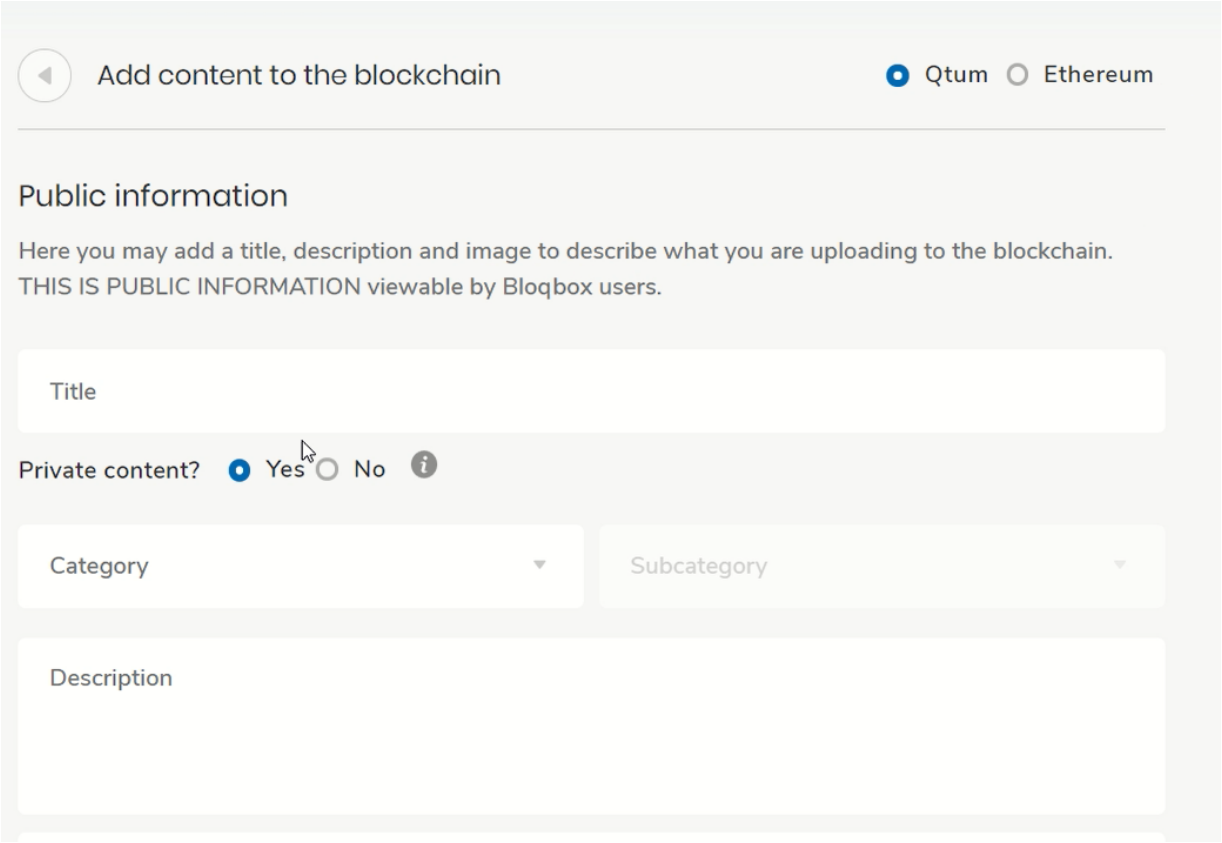


Рис. 3.2. Форма для заповнення контенту

Після натискання на кнопу «+» на (рис. 4.4), користувач може перейти до форми створення контенту з такими полями:

- Назва, яка буде відображатися на у заголовку контента.
- Кнопка-перемикач, яка означає чи буде контент доступний для продажу. Опцію «Yes» краще використовувати, якщо потрібно тільки для захисту авторських прав на контент. Опцію «No» краще використовувати, якщо контент передбачається для продажу.
- Категорія: музика, фотографії, програми тощо.
- Підкатегорія: чим точніше ви вкажете свою категорію, тим легше іншим користувачам буде знайти ваш контент.

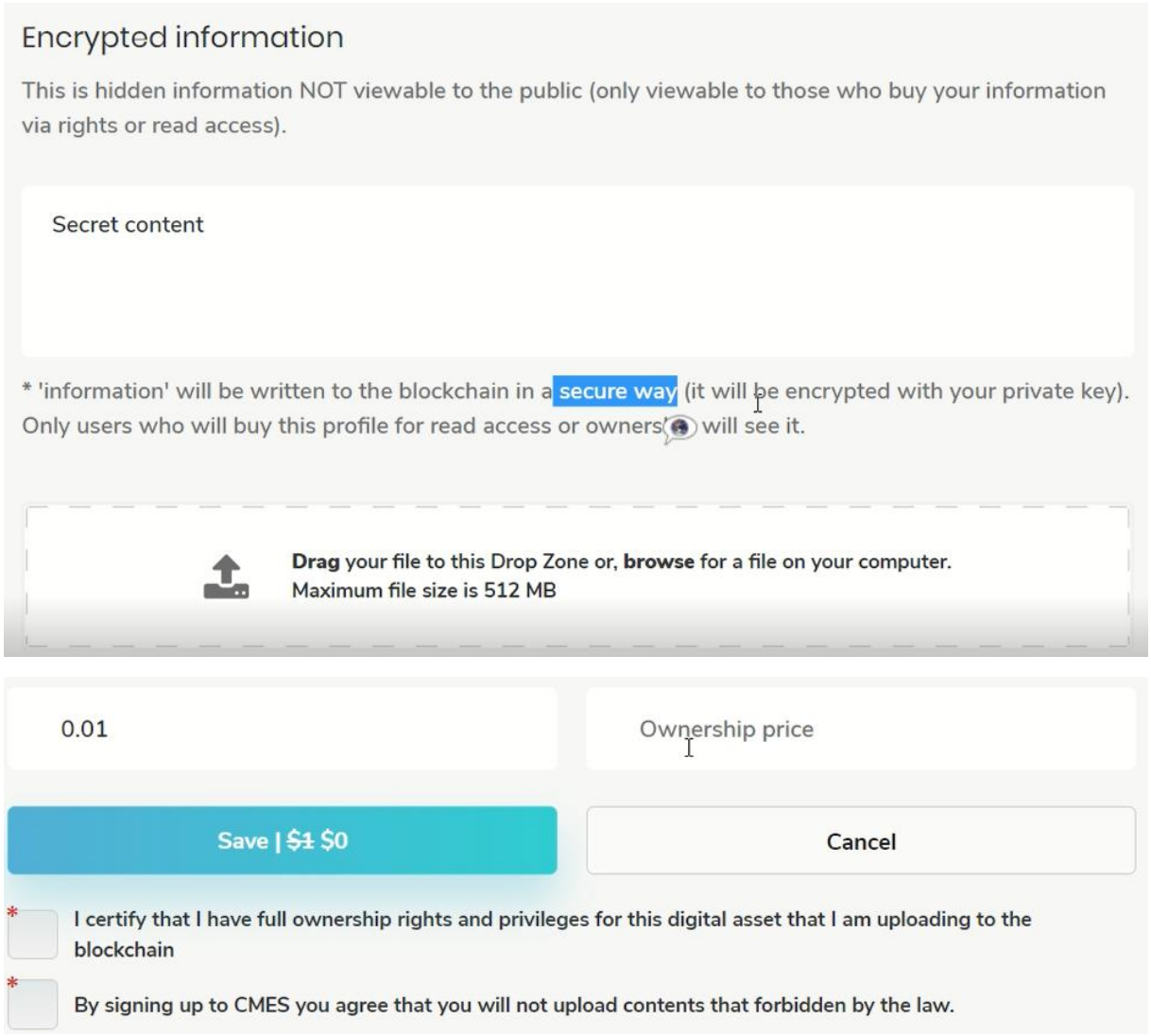


Рис. 3.3. Поле для вводу інформації про контент

Записати можна контент можна або у вигляді тексту або надати файл, який потрібно зашифрувати і записати в блокчейн. У другому полі потрібно перетягнути потрібний файл.

Можна виставити дві ціни:

– Ціна для читання – ціна, яку користувач повинен сплатити для того, щоб переглянит вміст файлу.
– Ціна для запису – ціна, яку користувач повинен сплатити для того, щоб мати право редагувати контент, змінювати самі ціни, а також продавати від свого імені.

Для того, щоб записати контент до блокчейну, користувач повинен погодитися із привилами системи:

– Права на контент належать йому.

– Вміст контенту не протирічить вимогам законодавства.

Після натискання кнопки «Save», контент записується на блокчейн і його можна побачити у вкладці «My contents».

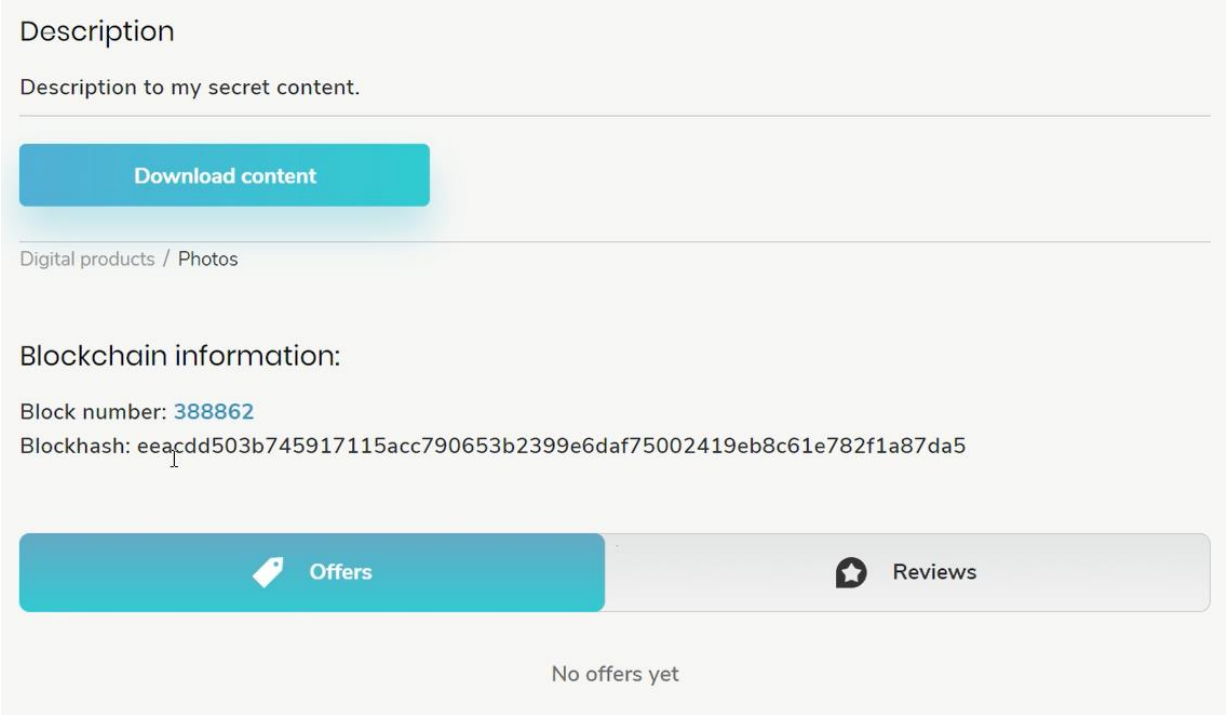


Рис. 3.4. Сторінка контенту

4. Купівля та продаж контентів

У сторінці контенту можна переглянути відповідну блокчейн інформацію, таку як номер та хеш блоку. Також відображаються офери на покупку і відгуки.

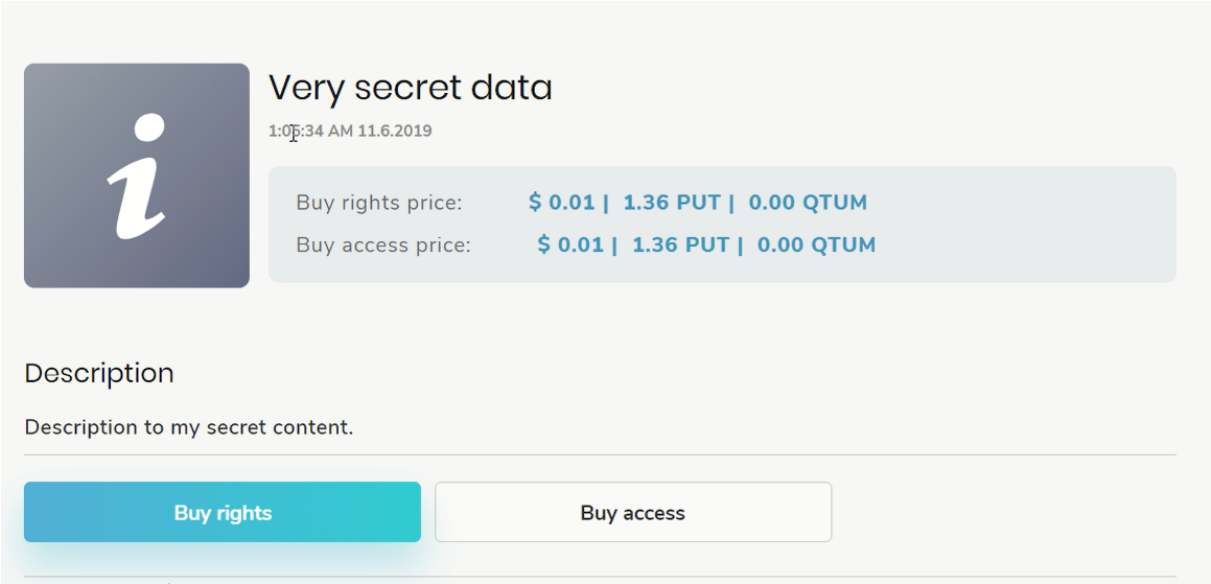


Рис. 4.1. Сторінка контенту для покупки

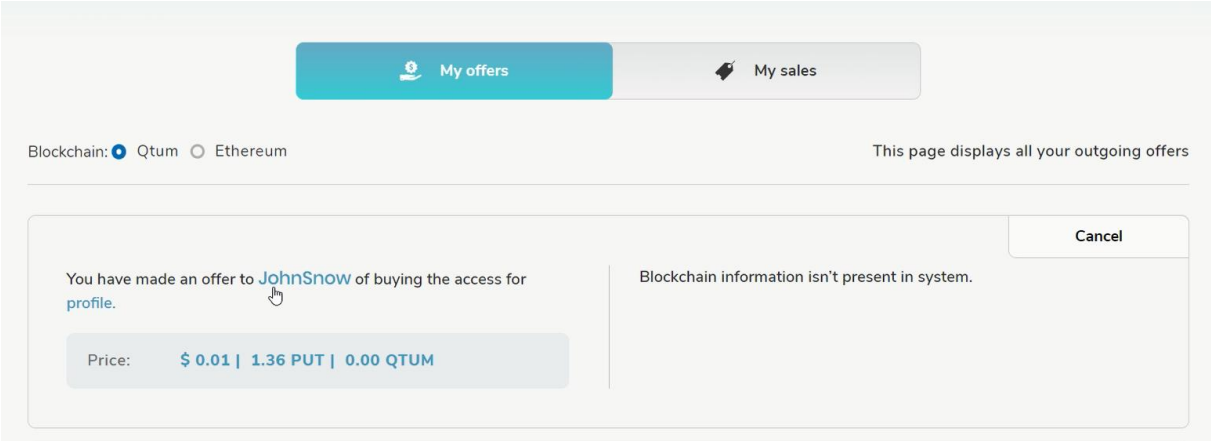


Рис. 4.2. Сторінка із оферами

Після того як було зроблено офер, покупець зможе переглянути його на вкладці «My offers». Справа, після того як транзакція підтвердиться, показується інформація про номер блоку та його хеш. Продавець у свою чергу зможе його побачити на вкладці «Invoices». Якщо покупець передумав покупати контент, він може натиснути на кнопку «Cancel», звісно це потрібно зробити до того як продавець підтвердить цей офер.

5. Редагування особистої інформації

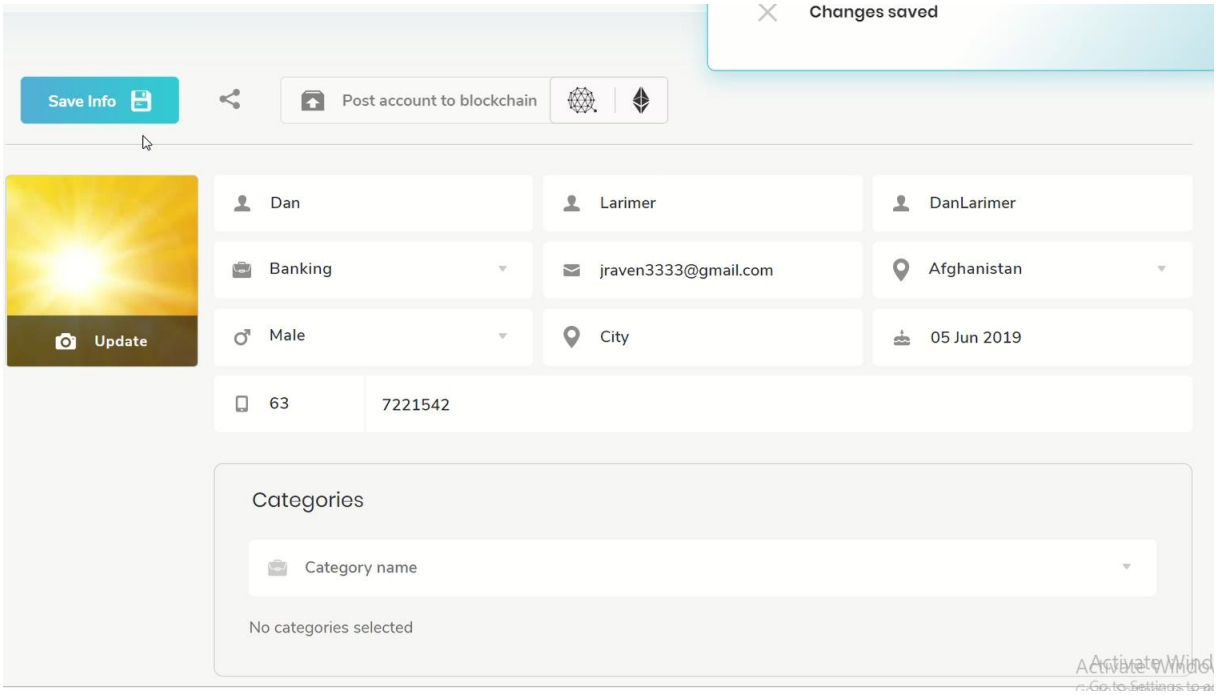


Рис. 5.1. Сторінка редагування профайлу

Після натиснення на кнопку «Feed», користувач має можливість змінити інформацію про свій провайл.

– Прізвище
– Ім’я
– Нікнейм
– Країна
– Поштова скринька
– Місто
– Дата народження

Також є можливість змінити аватар.

6. Поповнення особистого рахунку

Свій баланс, я також адреси для поповнення можна дізнатися при натисканні на кнопку «Deposit»

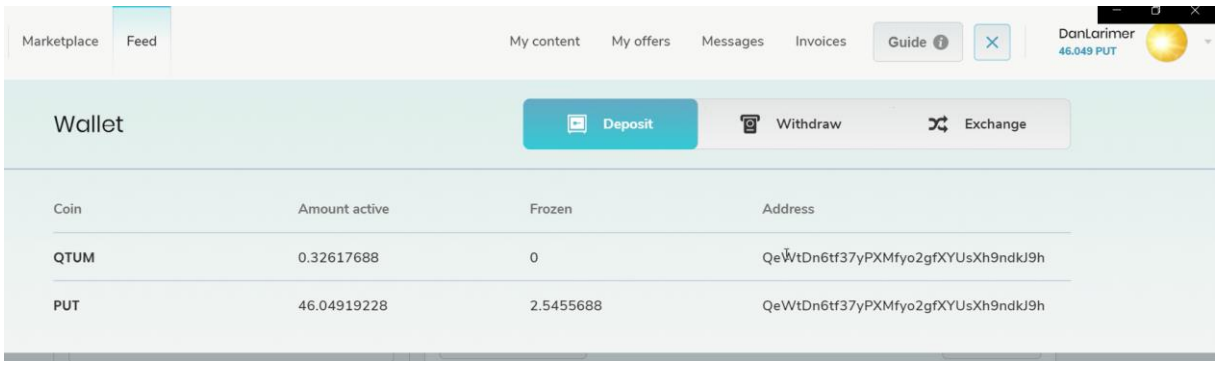
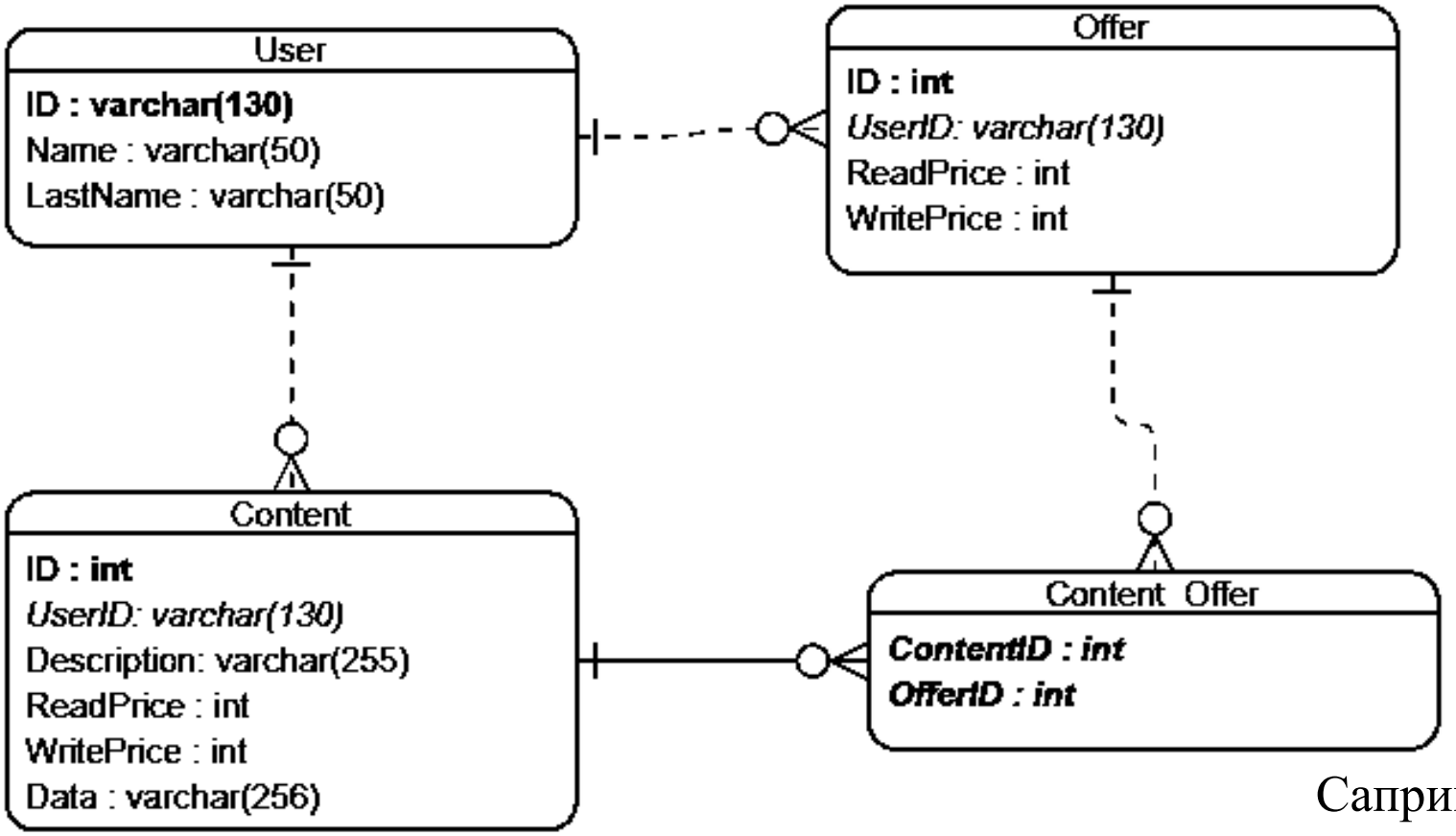


Рис. 6.1. Сторінка редагування профайлу

Справа у колонці «Address» наведено адреси, на які потрібно переказати токени для того, щоб використовувати їх у системі.

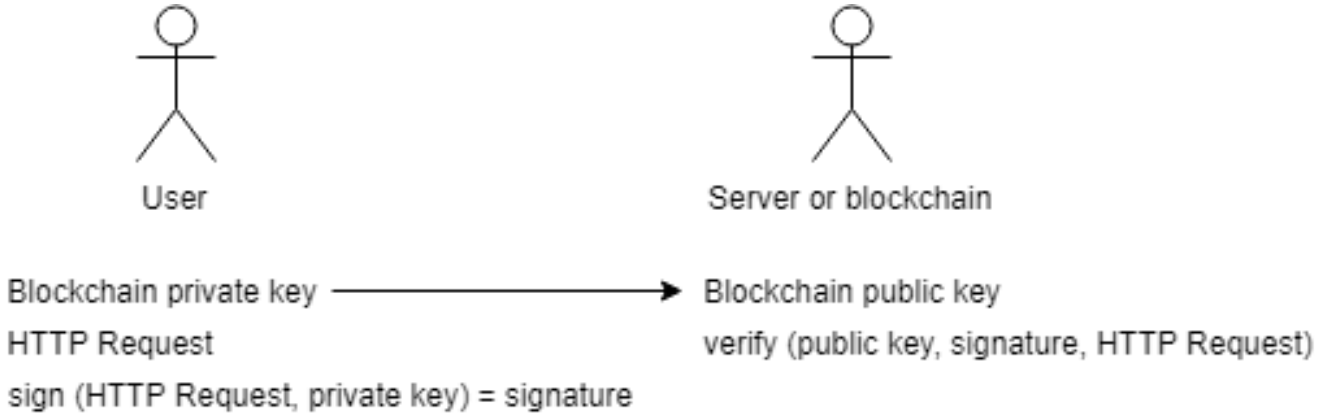
СХЕМА БАЗИ ДАННИХ



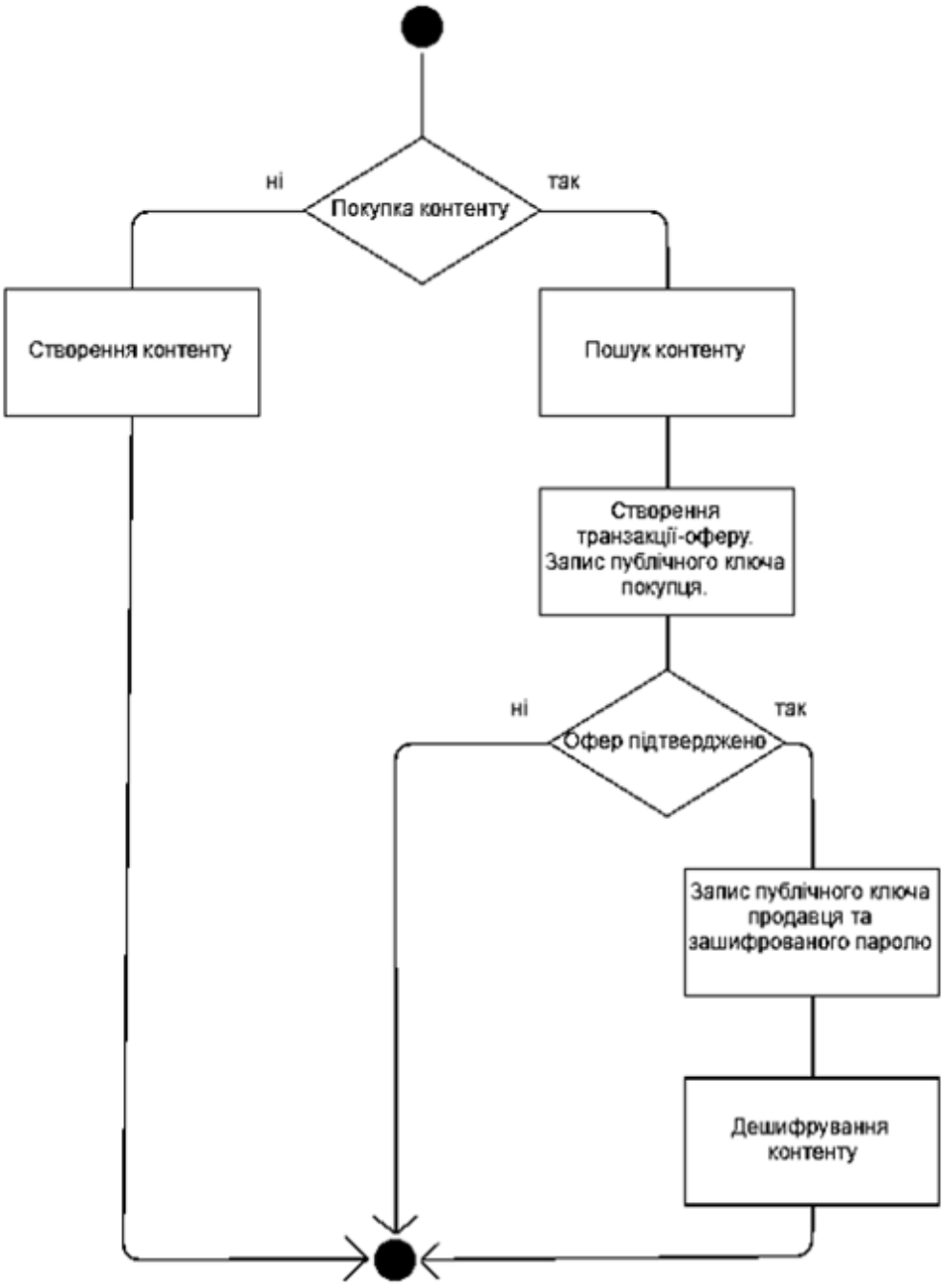
Саприкін А.О.

група КП-52

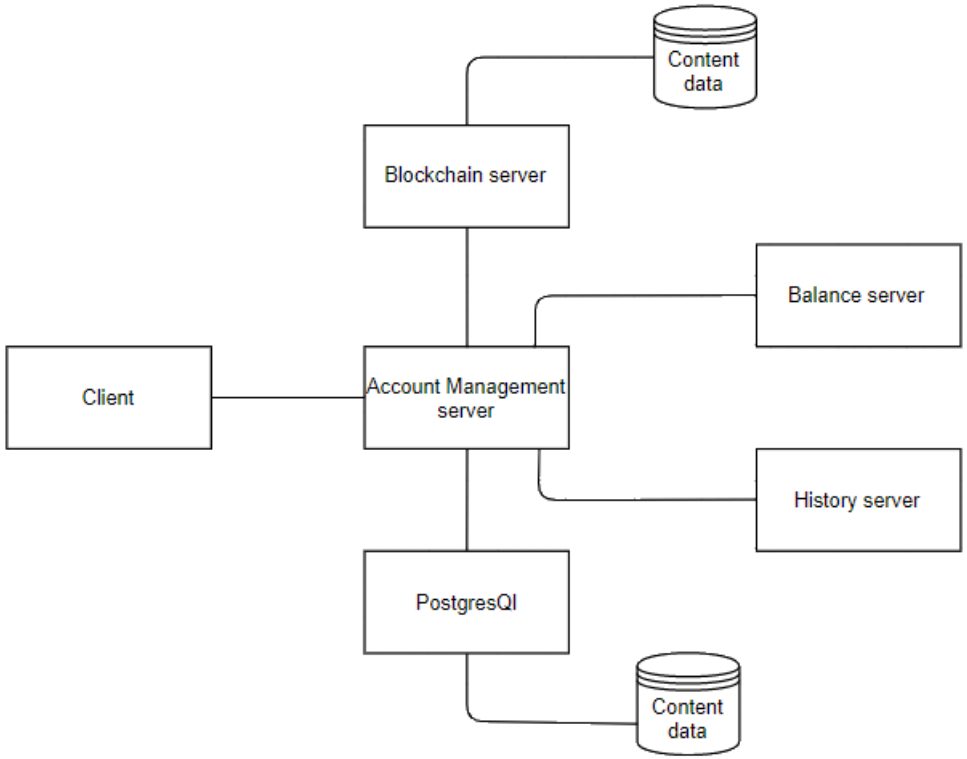
АЛГОРИТМ АВТОРИЗАЦІЇ



Саприкін А.О.
група КП-52



ДП.045440-07-99
Програмне забезпечення для захисту авторського права на основі технології блокчейн. Алгоритм передачі прав на контент. UML-діаграма



ДП.045440-06-99

Програмне забезпечення для захисту авторського права на основі технології блокчейн. Архітектура системи. UML-діаграма